Collaborative Cellular Tail Energy Reduction: Feasibility and Fairness

Girisha De Silva
School of Computing, National University of Singapore
girisha@comp.nus.edu.sg

Binbin Chen
Advanced Digital Sciences Center, Illinois at Singapore
binbin.chen@adsc.com.sg

Mun Choon Chan
School of Computing, National University of Singapore
chanmc@comp.nus.edu.sg

ABSTRACT

While cellular interfaces are known to be main power consumers in smartphones, our measurements reveal that surprisingly, due to cellular tail effect, a substantial ratio of energy drain comes from the low-frequency, low-data-rate background traffic. To reduce cellular tail energy use, existing solutions either require changing application behavior or assume the availability of low-level control of cellular interface (e.g., fast dormancy). We propose a collaborative cellular tail energy reduction approach for background traffic, which opportunistically discovers neighbours using low-power Bluetooth radio and shares their cellular bandwidth. Our evaluation demonstrates up to 90% saving for background traffic energy consumption in urban settings. Furthermore, we define a rigorous fairness notion by adapting the generalized processor sharing concept. Our sharing scheduling algorithm uses this notion to achieve substantial energy saving while ensuring fairness for participating phones.

CCS Concepts

•**Networks** → **Mobile networks;** *Network protocol design; Network resources allocation; Network simulations; Network measurement;*

Keywords

Cellular Networks, Energy Savings, Bluetooth Low Energy, Fairness, Resource Sharing

1. INTRODUCTION

Mobile data will continue to grow at an unprecedented speed in the rest of this decade. As predicted by Cisco Systems, mobile IP traffic will reach an annual run rate of 190 exabytes by 2018, as compared to 18 exabytes recorded in year 2013 [1]. Supporting this fast expansion poses significant challenges at both ends of the wireless links, in particular, the battery resource on mobile devices and the computing / communication resources of cellular infrastructure.

To conserve resources both on mobile devices and on base stations, a mobile device releases its reserved data channel and goes into an IDLE state after finishing its data transfer. However, frequent setup and teardown of cellular data channels can result in both significant delay for serving new traffic and in high consumption of base station resources (in terms of both computation and signalling channels). To avoid this, the standard cellular network design is to let a cellular interface stay in an intermediate state for a certain duration (*cellular tail phase*) after it completes the last data transfer. With the underlying correlation of traffic arrival time, there is higher probability that new data transfer may occur within this tail phase. When this happens, the tail phase helps avoid both the setup delay and the associated resource expenses. While cellular tail phase design has proved to be useful, it comes with its own downsides. As reported in [2], there are cases that cellular tail energy consumption can contribute to more than 60% of the total communication energy consumption.

Existing solutions to mitigate the wastage of cellular tail phase typically involve some traffic prediction mechanisms. Based on the prediction, the phone can terminate the tail duration earlier using the fast dormancy mechanism ([3, 4, 5]). Another popular approach is to batch several data transfers originated from a mobile device's different applications ([5, 2]). Such a technique aims to schedule the data transfers so that they share a common tail rather than incurring their individual tail phases. For these approaches to be effective, they either assume the capability to change/predict the behaviour of applications or the availability of low-level control for cellular interface (e.g., an API call to execute fast dormancy).

A new focus: background traffic. Our approach to mitigate the tail energy consumption is unique in that instead of looking at application data, we focus on the low-data-rate background traffic. Many mobile applications such as social media applications, email & calendar applications, and stock market applications generate background traffic to check for data update from their respective servers via the Internet. Smartphones typically spend substantial amount of their time with their screen off [6]. Background traffic continues to happen even during these periods in order to alert the user with updates. Hence, while their update frequency and data rate are low, the existence of tail phase for background traffic can make their aggregated impact on the battery lifetime rather significant.

As an illustrative example, suppose that the duration of the cellular tail phase is $20s$ and the average power consumption in the tail state is $600mW$. If a heartbeat message is transmitted in the background every $100s$, even without any other communication, the total energy consumed by the incurred tail phases in a 12 hours recharging period is $600mW \times \frac{12h \times 20s}{100s} = 1.44Wh$, which is around 26% of the total battery capacity ($5.45Wh$) of iPhone 5. If the user intends to use the phone longer, the ratio will further increase (e.g., greater than 50% for use of 24 hours).

ICDCN '16, January 04 - 07, 2016, Singapore, Singapore

ISBN 978-1-4503-4032-8/16/01…$15.00

DOI: http://dx.doi.org/10.1145/2833312.2833451

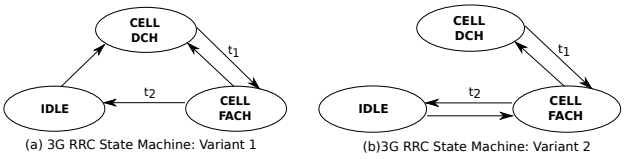


Figure 1: RRC state machines for 3G UMTS

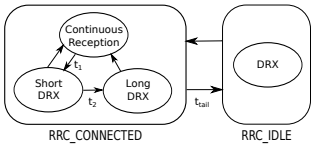


Figure 2: 4G RRC state machine

A collaborative approach: feasibility and fairness. Our solution is based on the observation that a smartphone often has a good number of other smartphones nearby and that each of these smartphones is equipped with both a high-power high-data-rate cellular interface and a low-power low-data-rate radio interface like Bluetooth. A *collaborative tail energy reduction* approach is thus feasible. Before turning on its cellular interface, a smartphone first opportunistically scans its neighbourhood using its low-power radio and checks whether a neighbor's cellular radio is already CONNECTED (including in its tail phase) and can help. When collaboration is possible, one device can utilize its low-power radio to transfer its background traffic via the helper phone and completely eliminate its own cellular tail energy. While the other device's tail duration may be extended, since the total amount of tail energy consumed is reduced, a fair scheduling algorithm can be used to ensure that all participating devices benefit from the collaboration over a long run.

Our evaluation based on real-world traces from urban environments demonstrates that this approach saves up to 90% of energy. We also validate the feasibility of our approach through a prototype implementation on Android phones. In order to ensure that such sharing benefit participating phones in a fair manner, we define a rigorous notion of fairness by adapting the well-accepted concept of generalized processor sharing [7]. Guided by this notion, we design a fair sharing scheduling algorithm that can achieve substantial energy saving while ensuring a satisfactory level of fairness.

Roadmap. The rest of the paper is organized as follows. Section 2 presents the background on radio state machines of cellular networks and the prior efforts on reducing cellular tail energy wastage. Section 3 presents our measurements that demonstrate the significant hidden cost of background traffic due to the cellular tail effect. We then reason about the feasibility of a collaborative tail energy reduction approach in Section 4. The fairness notion is defined in Section 5 and the realization of our collaborative cellular tail energy reduction is discussed in Section 6. We present results from both actual smartphones experiments and trace-based simulations in Section 7 and conclude in Section 8.

2. BACKGROUND AND PRIOR WORK

This section presents the necessary background on cellular tail phase and discusses the prior efforts on mitigating the energy wastage in cellular tail phases.

Radio resource management in 3G and 4G. Both 3G UMTS [8] and 4G LTE [9] are resource constrained systems. To manage the scarce resources efficiently, each mobile device, or so called User Equipment (UE), that accesses these network systems follows certain Radio Resource Control (RRC) state machine. This state machine determines the radio resource allocation for a UE in different states, hence it directly impacts the energy consumption and performance of UEs. State promotions (and demotions) result in signalling between the radio network and the UE, hence the frequency of promotions (and demotions) has a direct impact on the performance of a radio network.

Figure 1 presents two variants of the RRC state machines for 3G UMTS. The IDLE state in both variants refers to the case when a UE is switched ON but does not have an active data connection with the Radio Network Controller (RNC). This state consumes the least amount of power among the three states. A UE in the CELL_DCH state has a dedicated connection with RNC for data communication. This is the state with the highest power consumption rate. A UE in the CELL_FACH state is also connected to the RNC, but its connection is shared with other UEs, hence both the power consumed and the available data rate in this state are lower than the CELL_DCH state.

Once a data transmission is completed in the CELL_DCH state, an inactivity timer is started. If there is no more communication at the cellular interface for t_1 time, the UE is demoted to the state CELL_FACH and another inactivity timer is started. If there is no communication via the cellular interface for another t_2 time, the UE will demote to the IDLE state and the connection with the RNC will be terminated. We call the time period of $t_1 + t_2$ the cellular tail time. If the cellular interface has some traffic in the tail phase, the corresponding timer will be reset. When a UE is in IDLE state, in variant 1 (see Figure 1a) an activity triggers a UE directly into CELL_DCH state, whereas in variant 2 (see Figure 1b), it will first move to CELL_FACH state. Variant 2 is more energy efficient than variant 1 for low-data-rate transmission, albeit at the cost of an increased initial delay for supporting high-data-rate transmissions.

Figure 2 presents the RRC state machine for 4G LTE. Unlike the 3G UMTS RRC state machine shown in Figure 1, the 4G LTE state machine consists of only two main states. A UE in the state RRC_CONNECTED has dedicated radio resources for data communication, while a UE in the RRC_IDLE state does not. Similar to 3G UMTS, the completion of a transmission triggers an inactivity timer. If no more communication occurs for a period of t_{tail} time, the celluar interface is demoted to the RRC_IDLE state. Any communication that occurs before that resets the inactivity timer. Note that the RRC_CONNECTED state is itself a combination of 3 sub-states to allow a UE to carry out micro-sleeping for saving energy. More details on these sub-states can be found in [10]. We will present our power measurement results of different 3G/4G states in Section 7.

Prior efforts on reducing cellular tail energy wastage. There have been a substantial amount of research efforts devoted to mitigating cellular tail energy wastage. One main approach (e.g., see [3, 4, 5, 11]) is to understand the applications behaviour and cut the tail earlier when no new transmission is likely to happen. There are two potential downsides of this approach. 1) Since the cellular tail phase provides faster responses to possible data transmissions, such an approach bears the risk that a wrong decision can cause extra delay in communication. While researchers have developed various traffic prediction schemes to improve the decision accuracy, understanding the behavior of an ever-growing set of mobile applications can be technically challenging and also do not align well with the layering design of network stack. 2) All existing efforts assume that a phone can cut the tails through an underlying fast dormancy mechanism offered by the cellular interface [12]. However, as acknowledged in [5], the availability of an API to invoke fast dormancy mechanism remains uncommon on today's smart-

phones, probably due to a lack of OS support and more fundamentally due to the potential risk of exposing such a low-level control to app developers. One can envision that an abuse of such interface by applications may lead to serious performance and stability issues to cellular network infrastructure. In fact a recent document [13] discusses the impact of fast dormancy and as to why service providers are reluctant to enable it.

Another main thrust (e.g., [5, 2]) focuses on sharing the cellular tail across different applications to reduce the power consumed in tail phases. Specifically, if a phone can batch together data transfers originated by its different applications, these transmissions would be able to share a common tail phase rather than incurring their individual tail phases. Again, doing so requires a mechanism to change the default behavior of applications, which is not commonly available today.

The mobile *kibbutz* [14] is a system that targets to reduce the energy consumption and the latency associated with cellular networks via a mobile-to-mobile collaborative system using low power radios such as WiFi or Bluetooth. Kibbutz mainly targets relatively high data rate applications like audio streaming and web browsing whereas our proposed solution targets low-data-rate delay-insensitive background traffic. In order to improve the latency the kibbutz system needs to maintain the collaborative connections and therefore for low-data-rate traffic such as background traffic the energy overhead of keeping the connections alive dominates. In comparison, our proposed solution opens a Bluetooth connection only when a phone has data to forward or when its cellular interface is turned on. While this introduces some startup delay, it significantly reduces the energy consumption for keeping the Bluetooth connection. While the recent work by Hu and Cao [15] also considers the use of phone-to-phone connections to aggregate traffic and reduce energy wastage due to cellular tail, they do not focus on the background traffic. For foreground traffic, we find that using Bluetooth as the phone-to-phone interface incurs significant delay if the traffic volume goes beyond a few hundred kilo bytes, and using WiFi interfaces consumes significant energy on themselves thus limits the total savings. Finally, the *PhonePool* framework [16] attempts to reduce energy consumption by selecting the best cellular links among a set of collaborating phones. The sharing allows utilization of a better channel as well as the statistical reduction of tail energy. PhonePool focuses on choosing the best cellular link, hence reducing tail energy is only a secondary issue. For transfers with low data volume, the link rate plays a minor role, PhonePool's threshold-based algorithm reduces to choosing the link with the longest remaining time tail.

Our approach is unique in that we focus on the problem of low-frequency, low-data-volume and delay-insensitive background traffic. Our solution is based on a rigorous notion of fairness that derives from fair queuing context. We have also evaluated the energy efficiency of our approach using both simulation as well as a prototype implementation on Android phones.

3. THE HIDDEN COST OF BACKGROUND TRAFFIC

We approach the cellular tail energy wastage problem differently by focusing on background traffic that occurs even when a phone's screen is off. Somewhat surprisingly, our measurements show that a substantial amount of energy is consumed in supporting these low-frequency low-data-rate background traffic due to the cellular tail effect.

What goes on when your phone's screen is off? Many mobile applications, such as social networking applications (e.g. Facebook, WhatsApp) and email applications, use a push-based mechanism to efficiently receive their infrequent and aperiodic updates / notifications from their respective servers. The push-based mechanism works by first establishing a permanent TCP connection between a mobile device and a notification server. For example, Google provides such notification service for all Google applications that use the push-based mechanism. Once there is an update for any of the registered mobile applications, the corresponding application server will inform the notification server. This triggers the notification server to reach to the mobile application through the TCP connection between the notification server and the mobile device. Such a notification then triggers the mobile application to establish a direct connection with the corresponding application server to pull the content.

For contents that are updated infrequently and aperiodically, such a push-based mechanism is more energy efficient than a periodic pull-based mechanism, which requires smartphones to contact the application servers periodically regardless of whether there is any update. Although the push-based mechanism avoids the periodic pulling of content, in order to maintain the TCP connection between a mobile device and the notification server, the mobile application and the notification server still needs to send out heartbeat messages occasionally to prevent the TCP connection from timing out. A recent study [17] has found out that TCP connections behind cellular service providers' NATs (Network Address Translators) will be terminated if their idle period exceeds some time interval. The reported time intervals in [17], vary between 255 sec to 20 min. Since application developers need to design applications to run over the networks of different cellular service providers, they often conservatively send the heartbeat messages frequently enough so as to avoid such situations.

With the use of high data rate 4G-LTE networks, the total time to transfer small amount of data, like heartbeat messages, is negligible. Therefore, in a scenario where the only data transfers are exchanges of heartbeat messages, the energy consumption from the cellular radio will mostly be the power consumed by the cellular tail. When the frequency of heartbeat messages becomes sufficiently high, a large portion of the energy on the mobile device can be wasted by the cellular tail energy.

Background traffic and tail phase together kill the battery. We carry out measurements on off-the-shelf android devices to quantify the potential impact of background traffic. In our experiments, we run TCPdump on a Google Nexus 5 phone running Android version 4.4.4 (KitKat) to capture background traffic. The phone screen is switched off throughout the experiment. When processing the packet traces, we group together packets that arrive within a period of no more than 2 seconds. We consider each cluster of packets as a background event.

We tested four different settings. In the basic "Google Services Only" setting, the phone only runs the default Google applications and services, including the email and calendar applications GMail and Google Calendar. In the second setting (Application Set A), we installed the online social networking application Facebook, the instant messaging application Facebook Messenger, and the Internet news application BBC News (all using push-based mechanism), on top of the default Google Applications. In the third setting (Application Set B), we further added the microblogging application Twitter to the set of applications installed in Application Set A. In the last setting (Application Set C), we added to Application Set B two more applications: the instant messaging application WhatsApp and the VoIP application Skype. Both of these applications are expected to provide more real-time user interactions. We carry out each measurements for at least 6 hours.

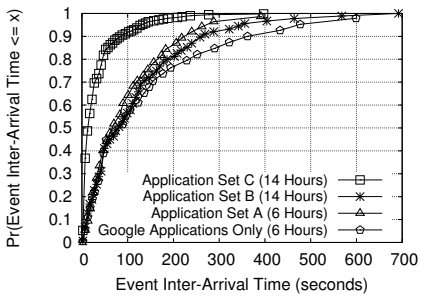


Figure 3: Distribution of background event inter-arrival time

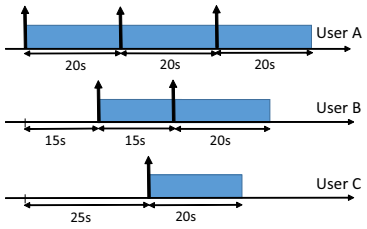
Figure 3 shows the inter-arrival time between background events under the different settings. The average inter-arrival time for the Google Services Only setting is roughly 140s. With the addition of new mobile applications, this reduces to around 100s for Application Set A. The inter-arrival distribution does not change much between Application Set A and B. This is most probably because both sets of applications use the Google Cloud Messaging service, which is a shared push-based service available through the Android API [18]. Using this underlying messaging service minimizes the number of connections to be maintained. Finally, the inter-arrival time for the setting of Application Set C is significantly lower than the application set A or B. The addition of the two real-time applications WhatsApp and Skype reduces the average inter-arrival time for application set C to around 30s. This suggests that the use of real-time applications results in a huge increase of background traffic. The size of the data exchanged in each of these background events ranges from a few hundred bytes to 10 KB. Further, we noticed that in very rare instances the background transfers were larger than 10 KB and these were removed from our traces. Previous studies have also reported [17] similar behavior for background transfer sizes.

Assuming a tail duration of 20s, we observe that a smartphone running Application Set A or B will stay in the tail phase for roughly $16-20\%$ of time. This ratio increases to 40% when running Application Set C. Over a 12 hour period, this constitutes around 25% of a 5.45Wh battery capacity (that of iPhone 5) for application set A and B, and around 50% for application set C. If the user intends to use the phone longer before the next charging, this ratio will further increase. Our measurement results clearly indicate that reducing the amount of energy drain for background traffic is of great utility in practice.

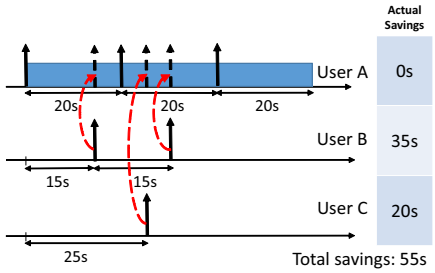
4. FEASIBILITY OF A COLLABORATIVE APPROACH

If we apply existing solutions (see Section 2) to deal with the tail phase energy wastage because of background traffic, we need to either 1) cut the cellular tails for background traffic, or to 2) change the behavior of the corresponding applications and try to group their background traffics together. As we discussed earlier, both approaches have their drawbacks. Fundamentally, they rely on support from other layers, i.e., an underlying fast dormancy API and applications cooperation respectively, which are not commonly available today.

Instead, we take a different *collaborative tail energy reduction* approach: Before turning on its cellular interface for background traffic, a smartphone first opportunistically scans its neighbourhood using a low-power radio (e.g., Bluetooth) and checks whether a neighbor's cellular radio is ON (or in tail state) and can help. If some helper is available, the phone can utilize its low-power radio



(a) Energy consumptions without collaboration



(b) Energy consumptions when User A helps the others

Figure 4: An example collaboration scenario

to forward its traffic to the helper phone, hence avoid the activation of its own cellular interface and the associated wastage of cellular tail energy.

Figure 4 illustrates an example of how this collaborative approach saves energy. In this example, the tail timer is 20s. As shown in Figure 4, User A has 3 packets, User B has 2 packets, and User C has 1 packet. Their cellular active periods overlap, and without collaboration their total active periods add up to $115s$. Under our proposed approach, when User B and User C have new traffic, they will search around using their Bluetooth radio and find that User A is available to help. If User A is willing to help (we will discuss this issue in Section 5), User B and C then communicate over their Bluetooth interface with User A so that User A can forward the traffic for User B and C. This allows User B and C to totally eliminate their cellular interface activation and the energy wastage in tail phases. As shown in Figure 4b, with the help from User A, the total active periods for the 3 phones reduce to $60s$, a nearly 50% reduction.

There are several underlying requirements for such a collaborative approach to become feasible:

1) *Performance-wise*, a low-power radio like Bluetooth can support the collaboration. Such radios only support a low data rate, e.g., less than 1 Mbps as measured on Samsung Galaxy SIII (I9305) phones. Fortunately, the low-data-rate nature of the background traffic allows the collaboration to happen over such low-power and low-data-rate radios. As shown in Section 3, the size of the data exchanged in each of these background events ranges from a few hundred bytes to 10 KB. A similar observation is also reported by [17]. In comparison, if one wants to forward a typical web page (e.g., $2MBytes$ large) over such a low-power low-speed interface, the delay will become intolerable (more than a dozen of seconds). Also, the small size of background traffic ensures that forwarding them will not affect the performance of the helper phone's own transmission.

2) *Energy-overhead-wise*, the use of a low-power radio like Bluetooth only introduces negligible energy overhead. Background data transfers sizes lie in the range of few 100 bytes to 10 KB and the energy consumed to transmit a typical 3 KB background traffic only consumes less than $0.5J$ of energy, or less than 5% of the total en-

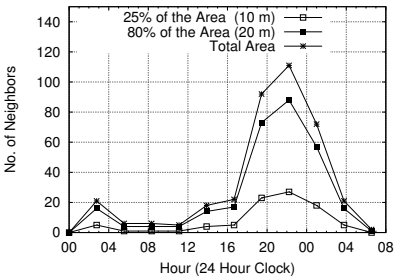


Figure 5: Distribution of number of neighbors in a food court

ergy ($12J$) wasted in a typical cellular tail phase. Furthermore, the helper phone (e.g., User A in the example of Figure 4b), turning on the Bluetooth interface for discovery and communication when the cellular interface (3G or 4G) is in high power state consumes nearly the same amount of energy as only turning on the cellular interface (see Section 7).

3) Collaboration opportunities are *sufficiently abundant*. If two phones wakeup for 20% of their time independently, the probability that they have some overlapping period will be low (around 4%). We observe that in urban settings (such as office, bus, metro, and food courts) a large number of smartphones stay close to each other for a considerable amount of time and therefore would provide an excellent environment for mobile-to-mobile collaboration. For example, on an average day an office worker who commutes to work via public transport will cumulatively spend at least 8 to 9 hours of his day in his office, at the food court/canteen and in public transport. Further, it is highly likely that this person would be meeting the same group of people everyday during his commute to work, at work and during lunch. Such a scenario provides an ideal situation for mobile-to-mobile collaboration.

In order to get a sense of the number of neighbors in an urban setting, we conducted an experiment to count the number of unique mobile devices using a WiFi sniffer at a popular food court. The reason to base our experiment on WiFi was that we found most of the mobile devices by default switch OFF the Bluetooth radio or keep it non discoverable. Counting based on Bluetooth will thus have very low coverage. Instead, we choose to count the number of WiFi devices. Note that while counting based on WiFi has better coverage, we are still underestimating the actual number of mobile devices since some mobile devices may still turn off their WiFi. If we consider Bluetooth to have the same range as WiFi, as shown in Figure 5, a smart phone would have more than 70 - 110 neighbors during the peak period (1900 hrs to 0100 hrs, for dinner and late-night supper) of the food court. The typical communication range of Bluetooth [19] is around 10 m, which was roughly 25% of the area covered by our WiFi sniffer. With that range we see roughly about 20 neighbors during the peak hours. Furthermore, our experiments in open spaces such as food courts and offices showed that a Bluetooth device can be discovered up to a range of 20 m, which is roughly 80% of the area covered by the sniffer. With this range roughly 50 - 90 neighbors can be discovered during peak hours. In terms of neighbors in an office space a recent study [20] reports that a typical office space of 100 m^2 (range of Bluetooth) houses around 10 - 15 people. If a node has 20 neighbors around, even if each neighbor independently wakes up only 10% of the time, the probability that there is at least one neighbor around to help is as high as $1 - (1 - 0.1)^{20} \approx 88\%$. Bluetooth devices can form a piconet of up to 8 active peers and up to 255 inactive peers [19]. Since only phones with background traffic needs to be active members, Bluetooth is suitable for supporting collaboration in a dense network.

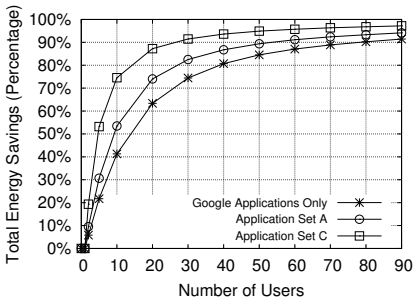


Figure 6: Energy savings under varying community size

Besides ours, several other research efforts [21, 14, 16] also depend on the availability of mobile-to-mobile collaborative opportunities.

Potential savings under varying neighborhood size. To further illustrate the last point above, we will do a more careful evaluation of the potential energy savings through collaboration under varying neighborhood sizes. For this, we pessimistically assume that no phones in the system are in active use. Having active phones around will increase the opportunity that a phone with background traffic can find a helper. We use the background event trace we collected (see Section 3) to drive the simulation. We simulate a clique of phones with an increasing neighborhood size. We evaluate the fraction of the total amount of energy that can be potentially saved through collaboration. As in Figure 4, we assume that if a phone with a new background traffic event finds another phone with active cellular connection, the latter will always help. This means that at any moment, for a clique of phones, there will be at most a single phone with active cellular interface.

As shown in Figure 6, based on our trace, as long as a node has more than two neighbors, the potential energy savings for the community can be around 5% and 10% respectively when the phones are running the default Google Applications and Application Set A. Further, around 20% can be saved if the phones are running Application Set C. Remember that in Application Set C, WhatsApp and Skype applications send background traffic more often to check for updates, hence increase the chance that two phones have overlapping periods. Even with a small neighborhood size (< 5 phones around), the energy saving is already sufficient to compensate for the overhead of Bluetooth probing. As soon as the neighborhood size reaches 10, the potential savings increases to more than 40% and around 75% respectively for Google Applications Only / Application Set A and Application Set C. This further increases to more than 60% and nearly 90% when the neighborhood size increases to 20. The rate of increases drops with further increase of neighborhood size. Our simulation result clearly shows that with a reasonably large neighborhood size (e.g., 10 to 20), which are already commonly available today, a collaborative approach has the potential to save a significant fraction (more than 50%) of the total energy wasted by background traffic in their tail phases.

5. FAIRNESS IN COLLABORATIVE ENERGY REDUCTION

In the previous section, we show that it is feasible to save significant amount of energy using a collaborative tail energy reduction approach. In the example of Figure 4b, the most energy efficient sharing is achieved by letting User A serve the other two users. However, under such an arrangement, User A performs all the works without gaining any savings from the collaboration,

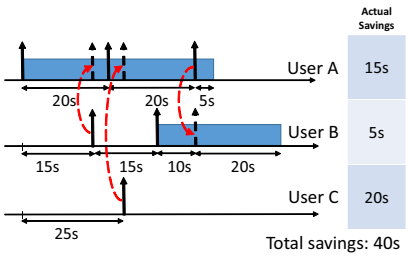


Figure 7: Energy consumption with Users A and B contribute

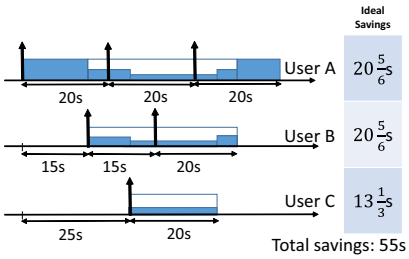


Figure 8: An ideal scheme with fluidly splittable energy savings

while Users B and C do not utilize their cellular radio at all. This is obviously "unfair", despite that it maximizes the total amount of energy saved.

Figure 7 shows another arrangement, where User A serves only a limited amount of time (in this example, until serving User C's event at $25s$). Since we do not assume a phone can cut its tail, User A will continue to be in an active state until $25 + 20 = 45$s. Since User A does not help with the second event of User B that arrives at $30s$, User B turns on its own cellular radio for that. User B will also help User A with A's last event at $40s$. User C still does not utilize its cellular radio at all. In this case, both Users A and B contribute. Through the collaboration, Users A, B and C reduce their cellular active time by $15s$, $5s$ and $20s$ respectively. Intuitively, this schedule would seem to be fairer than the greedy schedule in Figure 4b, where an active user will always help others in order to greedily reduce the total energy consumption. Achieving this improved fairness however comes at a cost. The total saving is now $15s + 5s + 20s = 40s$, which is smaller than the total savings of $55s$ in the greedy schedule. This lost of efficiency is because during the period between $30s$ and $45s$, both User A and User B turn on their cellular interface to switch the role of being a helper.

Based on the above examples, it is clear that what is needed is a quantitative definition of fairness. Based on a well-defined fairness notion, one can then evaluate different collaborative algorithms by their abilities to achieve both energy efficiency and fairness.

Ideal / perfectly fair savings. To establish a solid fairness notion in our unique collaboration context, we draw inspiration from the deep literature of packet scheduling. One widely-adopted fairness notion is the *Generalized Processor Sharing (GPS)* [7], which is a scheduling algorithm that achieves *ideal fairness* among multiple flows by assuming that the traffic is fluidly splittable. Under GPS, each flow can instantly receive its ideal fraction of the server at any point of time. Instead of the fair resource to be allocated, in our context, we consider the amount of *savings* that a mobile phone is entitled to. Similar to GPS, we define the *ideal (or perfectly fair) savings* for a mobile phone by assuming that at any point of time the potential amount of energy savings (as compared to a non-collaborative setting) can be fluidly split among all collaborating phones. Specifically, consider a moment when there are n phones that are in the cellular active states if they do not collaborate. Suppose all phones can hear each other. If they collaborate to save energy, only one phone needs to enter the cellular active state. Hence the energy can potentially be saved by a fraction of $1 - \frac{1}{n}$. Our fairness definition assumes that this potential savings can be equally split among all nodes instantaneously, hence the ideal saving of a node at the considered moment is $1 - \frac{1}{n}$ of its normal power consumption.

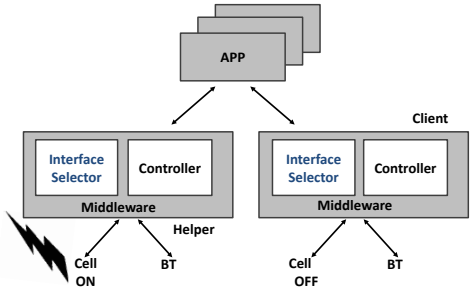


Figure 9: Proposed system architecture

Ideal saving is greater than zero only in periods when there are at least 2 nodes in cellular active states if they do not collaborate. Hence, in Figure 4a, saving is only possible in the period between $15s$ to $50s$. In this period, the amount of ideal saving varies with the number of active users. As shown in Figure 8, the ideal savings for both Users 1 and 2 are $10 \times \frac{1}{2} + 20 \times \frac{2}{3} + 5 \times \frac{1}{2} = 20\frac{5}{6}$. The ideal savings for User 3 is $20 \times \frac{2}{3} = 13\frac{1}{3}$. Note that as computation is based on what is possible when there is no sharing/collaboration, it is independent of the collaboration scheme used and captures the maximum amount of savings possible through collaboration.

With the notion of ideal fairness defined, we can then use it to gauge the fairness under different helper selection algorithms. One would expect that a good helper selection algorithm could let each node approximately obtain its ideal savings. However, this turns out to be impossible in general for our context due to a fundamental trade-off between efficiency and fairness in our context. Recall the example shown in Figure 7, the system has to let User A and User B stay in their cellular active state together for a while in order to achieve better fairness. Such an overlapping of cellular active phases reduces the overall system efficiency, making it impossible for all phones to achieve their ideal savings that is defined by assuming the ideal efficiency.

RMS fairness index. With this in mind, a natural goal is instead to hope that the ratio between a node's actual savings and its ideal savings is made similar across different nodes. The rationale behind this is that the inefficiency due to enforcing fairness should be shared among all nodes. Consider a set S of n phones, all with non-zero ideal savings. Denote phone i's actual savings and ideal savings by $actualSavings_i$ and $idealSavings_i$ respectively. We will evaluate the fairness of a scheduling algorithm by the degree that it can make the ratio of $r_i = \frac{actualSavings_i}{idealSavings_i}$ for different phones equal to each other. Specifically, we define the global average ratio $R = \frac{\sum_{i \in S} actualSavings_i}{\sum_{i \in S} idealSavings_i}$, and we define the *RMS fairness index* by the RMS value of the distance between r_is and R, i.e., $\sqrt{\frac{\sum_{i \in S}(r_i - R)^2}{n}}$. Note that for an ideal scheduling algorithm (as shown in Figure 8), the resulted RMS value is 0. Based on this definition, the two schemes as shown in Figure 4b and Figure 7 has a RMS value of 0.76 and 0.53 respectively.

In general, a scheme that is fairer tends to be less efficient, and vice versa. In the next section, we will present a scheme that provides a good tunable trade-off between these two requirements.

6. REALIZING COLLABORATIVE ENERGY REDUCTION

6.1 Prototype Implementation

Figure 9 shows the system architecture of our proposed system. Applications that make use of our collaborative system make all network related system calls via the middleware. The middleware has two components, an interface selector and a controller. The proposed algorithm, to be presented in Section 6.2, runs in the controller.

The interface selector may transmit the packets received from the applications either to the Bluetooth or the 3G/4G interface depending on the decision of the controller. A node can either be in the default or collaborative mode. In the default mode, packets are routed based on the default configuration.

In the collaborative mode, a node can be either a helper or a client. As a client, a node turns off its cellular interface. It will transmit and receive all traffic to/from the Bluetooth interface, using the helper as the relay node. The helper turns on its Bluetooth interface to communicate with one or more clients. It will also turn on its cellular interface to support communication with the Internet for both its own and the client's traffic. For a client to route its packet through the helper, it has to encapsulate the packets using a special header. The helper node forwards these packets to a proxy that will interpret the header information and forwards the packet to the proper destination. This encapsulation is also performed in the reverse direction. Our mobile-to-mobile collaborative system architecture is similar to the multipath-TCP-based implementation presented in [14], although the system in [14] is designed for collaborations involving larger data transfers.

In our system, the collaborative mode is only enabled when the frequency and amount of data traffic is low, typically when there is only background traffic. Starting from an initial state (as a client with no associated helper), a node uses the Bluetooth service discovery protocol to discover if any mobile device in vicinity is interested in collaboration. In order to reduce energy consumption, such discovery is performed only when a node needs to turn on its cellular interface for communication. If collaboration is enabled through the discovery, the Bluetooth MAC address of the devices discovered are cached for later use.

A prototype of the system was implemented on Samsung Galaxy SIV (LTE) phones running Android 4.4.2 (kitkat). We will use this prototype implementation in our evaluation in Section 7.

6.2 Algorithm Description

Based on the framework of fair queuing [22] and processor sharing [7], we propose a *Fair-Ratio* algorithm that runs in the controller. The algorithm keeps track of two variables, *actualSavings* and *idealSavings*. Based on the exchange of state information through the Bluetooth communication and by simulating the behavior of each phone when there is no sharing, a phone can derive the parameter *idealSavings*, which keeps tracks of the ideal savings (as in Section 5) a phone is entitled to. The phone then computes the difference between the energy consumption when there is no sharing and the actual energy consumption, which gives the other variable *actualSavings*. The outline of the Fair Ratio algorithm is shown in Algorithm 1.

In the algorithm, the subset of phones that can be a candidate to help must be the one that either is in cellular active (including tail) phase or has an active background event. If a phone is a newcomer (idealSavings is 0), we set its idealSavings to 1 to avoid the divide-by-zero error and its actual savings is set with 50% chance to +1 or -1. This makes the scheme appear neutral to newcomers on the average. For each phone i with background traffic or with active cellular connection, we compute its ratio $\frac{actualSavings_i}{idealSavings_i}$.

Algorithm 1 Fair-Ratio Algorithm (with threshold θ)

1: $S \leftarrow PHONES_{withTraffic} \cup PHONES_{cellularON}$;
2: Initialize $actualSum$ and $idealSum$ to 0;
3: **for each** $phone$ in S **do**
4: Get $phone$'s $actualSavings$ and $idealSavings$;
5: **if** $idealSavings == 0$ **then**
6: $idealSavings \leftarrow 1$;
7: Set $actualSavings$ to 1 or -1 with equal probability;
8: **end if**
9: $phone$'s $ratio \leftarrow actualSavings/idealSavings$;
10: $actualSum \mathrel{+}= actualSavings$;
11: $idealSum \mathrel{+}= idealSavings$;
12: **end for**
13: $avgRatio \leftarrow actualSum/idealSum$;
14: $helper \leftarrow phone$ of max $ratio$ in $PHONES_{cellularON}$;
15: **if** $helper$'s $ratio < avgRatio + \theta$ **then**
16: $helper \leftarrow phone$ of max $ratio$ in S;
17: **end if**

Table 1: Power measurements for cellular networks

RRC State	Power Measurement	
	3G/3.5G	4G LTE
IDLE	30 mW	30 mW
SEND (Upload)	1500 mW	2300 mW
RCV (Download)	1200 mW	2100 mW
TAIL	600 mW	600 mW

Next, we compute the average ratio $\frac{\sum_i actualSavings_i}{\sum_i idealSavings_i}$ for all such phones. If for the current helper, the difference between its ratio and the average ratio is not too large (less than the threshold θ), it continues to help. A larger threshold θ increases the chance the current helper continues to help hence can make the system more energy efficient. In contrast, a smaller θ can lead to more fair distribution of loads among phones, albeit at the cost of total energy efficiency. If there are multiple existing helpers that meet the criteria, the helper with the highest ratio is chosen. If no existing helpers meet the criteria, i.e., all helpers have contributed θ more than their fair shares, the algorithm will optimize for fairness now. Under this case, the phone with the largest ratio will be chosen since relatively it has gained the most from its past collaborations.

An active phone periodically updates its helper about its event activities and the helper then shares this information with other phones. Information update does not have to be frequent. Note that longer delays in these updates tend to reduce the number of helper changes, thus making the performance less fair and more energy efficient.

7. EVALUATION

7.1 Power Measurements

We used a monsoon power meter [1] to carry out a series of power measurements on a Samsung Galaxy SIII (I9305) phone running Android 4.1.2. We kept the screen off throughout our experiments.

The first set of our experiments measure the power consumption in different states of the RRC state machine for both 3G and 4G networks. We summarize the results in Table 1. Our second set of experiments measure the power consumed by a Bluetooth 4.0

[1] http://goo.gl/sjChCH

Table 2: Power measurements for Bluetooth 4.0

BT State	Power Measurement
SEND	290 mW
RCV	320 mW
CONNECTED	65 mW

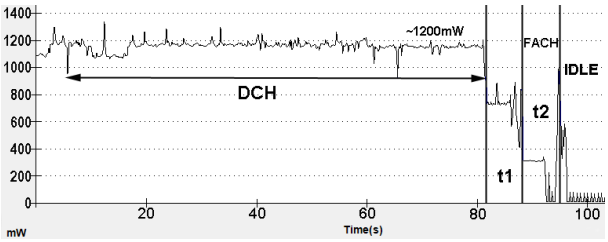


Figure 10: 3G cellular tail $\approx$ 10 seconds

radio, with the results summarized in Table 2.

We also measure the tail times for different cellular networks. The tail duration for a 3G network is roughly $10s$ (Figure 10). For two 4G networks operated by different service providers, we observe that the tail durations are around $10s$ and $20s$ respectively (Figure 11 and Figure 12).

Finally, we measured the effect of Bluetooth discovery when a smart phone is on its cellular ON stage. For this experiment, we first let the phone carry out low data rate transfers via the cellular connection at pre-defined intervals. Afterwards we averaged the cellular tail energy consumption since the cellular tail is the dominating fraction for low data rate traffic. In the next experiment we carried out the same low data rate transfers via the cellular interface while having the Bluetooth radio switched ON and advertising a Bluetooth connection. The difference between the average cellular tail energy consumption in the two experiments less than 10 mW.

7.2 Delay Measurements

Forwarding packets via Bluetooth through another device rather than through a device's own cellular connection incurs addition delay. We carried out an experiment to measure this additional delay. Our experimental setup consisted of two Android mobile devices where one of the phones (sender) forwards a 2 KB packet via Bluetooth 4.0 to the other phone (proxy) which has a 4G LTE connection. The proxy then forwards the packet (via 4G) to a remote server which will reply with an ACK packet to the proxy. This ACK packet will then be relayed from the proxy to the original sender via Bluetooth. For each packet/ACK exchange, both the sender and proxy keep track of the time of transmission and reception. We transmit 100 packets from the sender and measure the additional delay incurred by the extra forwarding through Bluetooth. The average additional delay measured is about 111ms, which is not significant for background notification traffic.

7.3 Trace-based Evaluation

We developed a customized simulator that incorporates the cellular RRC state machine of phones. The simulator also allows easy incorporation of each phone's network activity trace (i.e., when it sends and receives packets) and the phone-to-phone contact trace (i.e., in which time periods two phones become direct neighbors). We have made the source code of our simulator available [2]. Instead of using the collected traces described in Section 3 directly, we extract more than 1500 instances of event inter-arrival times as collected over 50 hours from the traces and randomly pick one of

[2]http://www.cir.nus.edu.sg/projects/tail-sharing

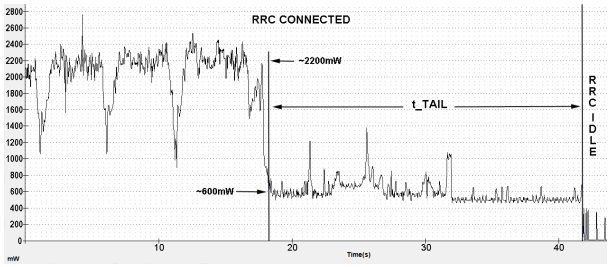


Figure 11: 4G cellular tail (Service Provider A) $\approx$ 20 seconds

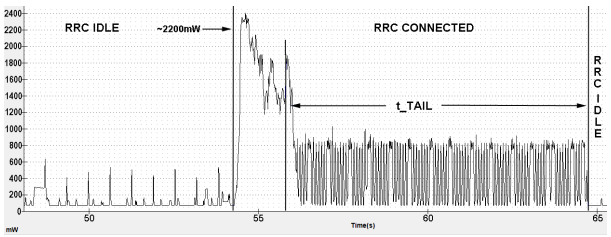


Figure 12: 4G cellular tail (Service Provider B) $\approx$ 10 seconds

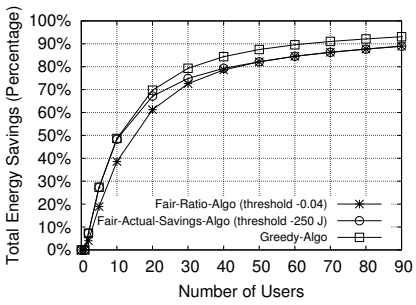


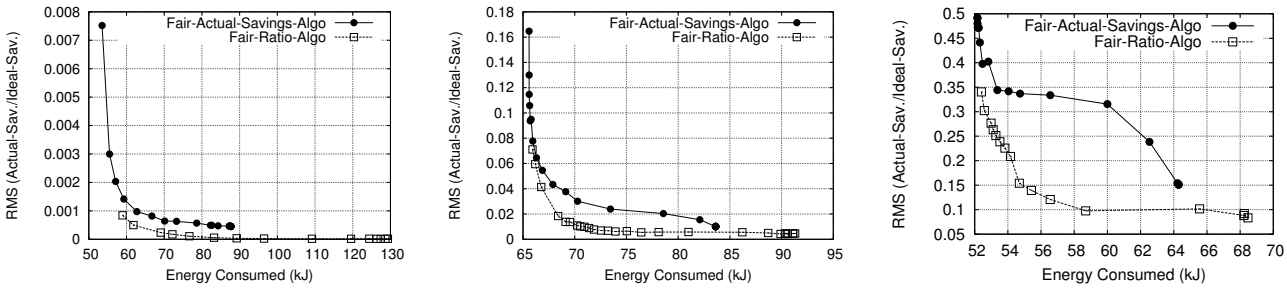
Figure 13: Energy savings under varying number of users

these inter-arrival times during simulation. Traces generated using this sampling approach spends on the average 16% of the time in active (cellular tail) state, similar to Application Set A and B described in Section 3.

In terms of power consumption, since packets for background traffic are typically small, the simulator incorporated a simpler 4G LTE RRC state machine where any packet transmission will trigger the mobile device to enter the tail power state, omitting the SEND/RCV high power state (RRC_CONNECTED). We also assume that the amount of energy consumed by Bluetooth 4.0 transmission is negligible.

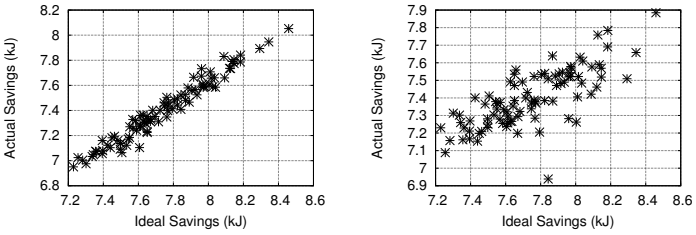
For contact patterns, we put a WiFi monitor in a food court to identify the presence of patrons through their WiFi packets. Data collection lasted for a period of 2 days. Using the collected user traces, we generated two contact traces. The first trace contains the top 100 longest occupancies (Dynamic trace) and each stay lasts for more than an hour. In the second trace (Mix trace), there is a mixture of 50% long (more than 1 hour), 25% medium (30min to 1hour) and 25% short (15min to 30min) occupancies. As some of the patrons visited the food court more than once, the Dynamic trace has 85 unique users while the Mix trace has 88 users.

The proposed Fair-Ratio Algorithm was evaluated against two other algorithms. One algorithm is a greedy algorithm where mobile devices would always pick (if available) the mobile device which has its cellular radio switched on. Hence, there should not be



(a) Static trace (b) Dynamic trace (c) Mixed trace

Figure 14: The trade-off between fairness and energy consumption for Fair-Actual-Savings-Algo and Fair-Ratio-Algo (simulates a period of 48 hours). As references, energy consumption for Static trace are: Greedy 52kJ, No-sharing 757kJ; for Dynamic trace: Greedy 66kJ, No-sharing 166kJ, and for Mixed trace: Greedy 52kJ, No-Sharing 113kJ



(a) (b)

Figure 15: Actual savings vs. ideal savings for the static trace:
(a) Fair-Ratio-Algo (RMS: 5.6×10^{-5}, Energy: 83.2kJ)
(b) Fair-Actual-Savings-Algo (RMS: 4.9×10^{-4}, Energy: 82.6kJ)

more than 1 device with its cellular radio on at any time. Although this greedy approach provides the most energy efficient solution, it is also the most unfair. The other algorithm is the Fair-Actual-Savings Algorithm. The Fair-Actual-Savings Algorithm picks the current cellular radio on device to collaborate only if its current gain is greater than the sum of the gain of all mobile devices that needs to collaborate plus some threshold. If no device meets the constraint, the phone with the highest gain within the set of phones that needs to collaborate will be selected. Note that while the threshold of the Fair-Ratio Algorithm is a ratio, the threshold for the Fair-Actual-Savings Algorithm is in Joule (J). We use the RMS fairness index as defined in Section 5 to measure the fairness.

Energy savings under varying number of users. We vary the number of users collaborating and measured the amount of overall savings. Users stay in the system throughout the simulation. As one can see in Figure 13, when the number of users increases, the amount of energy that can be saved increases in all three algorithms. Basically with more neighbors, there is higher probability of finding a phone to collaborate with and thereby reducing the energy consumption. Note that since the activity period is only 16% on the average, the opportunity for sharing when there are only 2 devices or users is rather small. However, even with only 5 users, a savings of almost 20% to 30% can be achieved. With 10 users, savings reach 40% to 50%. This shows that substantial savings is possible, even in an environment with say only 5 to 10 users. With 50 users, savings can be more than 80% and adding more users do not achieve much more gain beyond that.

RMS fairness index vs energy consumption. We compare the performance of Fair-Ratio and Fair-Actual-Savings Algorithm by varying the threshold and evaluate how they perform in terms of the trade-off between fairness and energy consumption. The energy consumption of the Greedy and No-Sharing cases provide the two extreme performance bounds, as detailed in the Figure 14 caption.

We first evaluate the performance using a scenario where there are always 91 users in the system. The result is shown in Figure 14a. Since the users are static and always present, we expect both algorithms to be fair and savings to be substantial. In fact, the RMS fairness index is less than 0.008 for Fair-Actual-Savings and less than 0.0009 for Fair-Ratio. With respect to No-Sharing, the total energy saving for Fair-Ratio Algorithm ranges from 83% to 92%. In addition, the Fair-Ratio Algorithm does not have any free riders in the range of thresholds evaluated. In comparison, the Fair-Actual-Savings Algorithm allows free riders when the energy consumption is less than 59kJ.

The result for the Dynamic trace is shown in Figure 14b. Again, in terms of the energy consumption vs. fairness trade-off, the Fair-Ratio Algorithm is better than the Fair-Actual-Saving Algorithm. The unfairness is higher since the variation in contact duration makes it harder to spread the gain fairly. The RMS fairness index varies from 0 to 0.16 for Fair-Actual-Savings and 0 to 0.07 for Fair-Ratio. Savings for the Fair-Ratio algorithm are smaller compared to the static case, ranging from 45% to 60% relative to No-Sharing. Unlike the static trace, when using the Dynamic trace, free riders exist for both the algorithms when the energy consumption is less 71kJ.

The results for the Mixed trace is shown in Figure 14c. The RMS fairness index is much higher since users spend very different amount of time in the system. The RMS fairness index reaches almost 0.5 in the worst case for Fair-Actual-Saving Algorithm. The Fair-Ratio Algorithm can still strike a good trade-off between the fairness and energy consumption. One interesting observation is that the curve for the Fair-Actual-Saving algorithm is different from all other settings. We believe that this is due to the large differences in gain between short contacts and large contacts and there is a large range of thresholds whereby there is loss in energy efficiency with no corresponding gain in fairness. Note that even larger threshold cannot bring the fairness further down in this scenario.

Note that the simulations in Figure 14 are based on a 48 hour mobility trace. The threshold values vary from -0.3 to +0.3 and from -800J to +800J for the Fair-Ratio algorithm and Fair-Actual-Savings algorithm respectively.

Per-user fairness. So far, the level of fairness is shown at an aggregated level. Figure 15 shows a scattered plot of the actual savings vs. ideal savings for individual user under both algorithms for the static trace. We select their respective thresholds such that their total energy consumptions are similar. The result shows the Fair-Ratio algorithm is able to maintain similar ratio for most users while the ratios of the Fair-Actual-Saving are scattered much more

Table 3: Prototype evaluation result

Setting	Data rate (bps)	Average events/s	Savings
App Set C	15	0.03	1.7%
C×5	72	0.11	7.6 %
C×10	127	0.18	11.8%

widely. As a result, in Fair-Actual-Saving scheme, some nodes' actual savings can unfairly exceed their ideal saving shares, while most other nodes' actual savings become lower than the Fair-Ratio scheme can offer.

7.4 Android Phone Prototype Evaluation

To validate the feasibility of our proposed approach on real mobile phones, we also developed a prototype system of our proposed collaborative tail energy reduction mechanism as described in Section 6.1.

Our prototype system consists of two Samsung Galaxy SIV smartphones with Bluetooth v4.0 as the low-power radio. Both phones are running Android kitkat (version 4.4.2). Each phone uses a subset of the trace from the application set C to mimic the background data traffic. Both phones use 4G LTE connections for Internet access, with the length of the tail period set to 10s by the cellular service provider. We measured the total energy consumption of the phones when executing our fair-ratio algorithm using monsoon power meters, and compared the results to a baseline where the phones do not collaborate. In order to overcome the shortage of power meters, we also use the two phones to mimic multiple phones, by let each phone independently sample multiple subsets from the trace of application set C. We use one phone to mimic 5, and 10 phones, and refer to the corresponding experiments as C×5 and C×10 respectively.

We run 3 experiments for each setting, and each experiment lasts for 1 hour. The first experiment only incurs background traffic of the phone. The second experiment incurs the background traffic and the phones execute our fair-ratio algorithm. The third experiment incurs the same background traffic (including our trace) without running any collaboration algorithm. Denote the total energy consumption of the three experiments by E_1, E_2, and E_3. We report the value of $\frac{E_3-E_2}{E_3-E_1}$ as the ratio of savings provided by our collaborative algorithm over the background traffic communication energy. Table 3 summarizes our experiment results, which demonstrates that our proposed collaborative energy reduction approach can reduce energy consumption on real mobile phones. It also confirms the trend that greater savings can be obtained with more participating phones. The saving here is lower than what is shown by the simulation, which is an artefact since all the virtual phones actually use the same cellular interface of the two underlying physical phones.

8. CONCLUSION AND FUTURE WORK

In this work, we have presented a new collaborative approach for mitigating the tail energy wastage in cellular communications. By focusing on background traffic, we devise schemes for nearby phones to share the overhead of their tail energy in an efficient and fair manner.

While these initial results are promising, the following issues worth further investigating: 1) With collaboration, a phone's traffic may go through neighboring phones and cloud-side proxies. This allows other phones to learn the presence of a phone and lets proxies to learn which services/apps runs on a phone. We plan to investigate practical mechanisms to address such privacy concerns. 2) While our approach aims to achieve fairness through scheduling, a credit-based method can be more flexible and suitable in highly mobile environments, though it also incurs new overheads. It is interesting to investigate how different mechanisms complement each other.

Acknowledgment

This research is partly supported by Singapore's Agency for Science, Technology, and Research (A⋆STAR) under the Human Sixth Sense Programme (HSSP) and partly supported by the Singapore National Research Foundation under its IRC@Singapore Funding Initiative and administered by IDMPO.

9. REFERENCES

[1] Cisco Systems. Visual networking index (VNI). http://goo.gl/n1iyMS. Accessed: 2014-07-24.

[2] N. Balasubramanian, A. Balasubramanian, and A. Venkataramani. Energy consumption in mobile phones: a measurement study and implications for network applications. ACM IMC '09.

[3] F. Yu, G. Xue, H. Zhu, Z. Hu, M. Li, and G. Zhang. Cutting without pain: Mitigating 3G radio tail effect on smartphones. IEEE INFOCOMM '13.

[4] S. Deng and H. Balakrishnan. Traffic-aware techniques to reduce 3G/LTE wireless energy consumption. ACM CoNEXT '12.

[5] F. Qian, Z. Wang, A. Gerber, Z. M. Mao, S. Sen, and O. Spatscheck. Top: Tail optimization protocol for cellular radio resource allocation. IEEE ICNP '10.

[6] J. Huang, F. Qian, Z. Mao, S. Sen, and O. Spatscheck. Screen-off traffic characterization and optimization in 3G/4G networks. ACM IMC '12.

[7] K. Parekh and G. Gallager. A generalized processor sharing approach to flow control in integrated services networks: The single-node case. *IEEE/ACM Trans. Netw.*, 1993.

[8] The 3rd Generation Partnership Project (3GPP). UMTS. http://goo.gl/NvPHQQ. Accessed: 2014-07-24.

[9] The 3rd Generation Partnership Project (3GPP). LTE. http://goo.gl/sOIxLk. Accessed: 2014-07-24.

[10] J. Huang, F. Qian, A. Gerber, Z. M. Mao, S. Sen, and O. Spatscheck. A close examination of performance and power characteristics of 4G LTE networks. ACM MobiSys '12.

[11] F. Xu, Y. Liu, T. Moscibroda, R. Chandra, L. Jin, Y. Zhang, and Q. Li. Optimizing background email sync on smartphones. ACM MobiSys '13.

[12] 3GPP Release 7: UE Fast Dormancy behavior, 2007. 3GPP discussion and decision notes R2-075251.

[13] 3GPP System impact of poor proprietary Fast Dormancy, 2009. 3GPP discussion and decision notes RP-090941.

[14] C. Nicutar, D. Niculescu, and C. Raiciu. Using cooperation for low power low latency cellular connectivity. ACM CoNEXT '14.

[15] W. Hu and G. Cao. Energy optimization through traffic aggregation in wireless networks. IEEE INFOCOMM '14.

[16] J. Lee, Yeongjin Lee, K., and S. Chong. Phonepool: On energy-efficient mobile network collaboration with provider aggregation. IEEE SECON '14.

[17] F. Qian, Z. Wang, Y. Gao, J. Huang, A. Gerber, Z. Mao, S. Sen, and O. Spatscheck. Periodic transfers in mobile applications: network-wide origin, impact, and optimization. ACM WWW '12.

[18] Android Developers. Google Cloud Messaging for Android. http://goo.gl/7N9WWZ. Accessed: 2014-07-24.

[19] Bluetooth Special Interest Group. Bluetooth technology website. http://www.bluetooth.com/. Accessed: 2014-04-16.

[20] The Sydney Morning Herald. How many workers can you fit in an office? http://goo.gl/48Ji87. Accessed: 2015-08-01.

[21] L. McNamara, C. Mascolo, and L. Capra. Media sharing based on colocation prediction in urban transport. ACM MobiCom '08.

[22] A. Demers, S. Keshav, and S. Shenker. Analysis and simulation of a fair queueing algorithm. ACM SIGCOMM '89.