ABSTRACT
Mobile devices today rely on cellular (3G or LTE) connectivity because it has ubiquitous coverage. Unfortunately, cellular links are energy hungry at low bit-rates and have high round-trip times after idle periods. These characteristics punish common mobile applications such as web browsing and streaming, decreasing battery life and user satisfaction.

We present the design and implementation of the Mobile kibbutz, a system that improves the performance of cellular links by having nearby users share their links with each other via shorter range wireless links - WiFi or Bluetooth. Without requiring application specific knowledge, instrumentation, or recompilation, the kibbutz allows multiple users' traffic to be consolidated on a subset of links, leading to reduced energy consumption and smaller round-trip times, while ensuring fairness across different users.

We have implemented the kibbutz on Samsung Galaxy Nexus devices and shown that for typical usage applications, collaborating phones consume 25% less energy when listening to radio, experiencing more responsive web browsing (1.2× faster on average), and 33% faster downloads.

Categories and Subject Descriptors
C.2.1 [Computer-Communication Networks]: Network Architecture and Design—Wireless communication

Keywords
3G; 4G; WiFi; Bluetooth; MPTCP; tethering; power

1. INTRODUCTION
Mobile devices have become more numerous than the traditional desktop computers, and much of their success is due to the “always-connected” paradigm they enable: users are constantly in touch with the latest news, receive email, navigate and chat while on the move. Cellular technologies such as 3G and LTE are the default choice of connectivity because they have ubiquitous coverage. Unfortunately, they are both energy-hungry and have high latency after periods of inactivity. Indeed, many mobile user forums single out 3G as the usual suspect for high battery drain and suggest turning it off when battery is low; LTE draws even more power than 3G.

Cellular networks rely on centralized capacity allocation for high utilization of the wireless spectrum and bandwidth and fairness guarantees that are not available in distributed access control such as CSMA. The price paid is the need for signaling between the mobile devices and the network: whenever a device needs to send or receive a non-trivial amount of data, it asks the network for a dedicated channel. Once the channel is allocated, the mobile device draws a lot of energy regardless of its data rate. Signaling is expensive both for the operator as it limits scalability, and for the mobile device, as it increases energy cost and communication latencies. That is why the mobile only releases the channel when there is no traffic during the tail timer, a parameter set by the MNO (mobile network operator).

Cellular connections offer a fundamental tradeoff between low latency on one hand, and low energy consumption on the other: we could achieve low latency if the mobile link were always on, at the cost of quickly draining the device battery. Mobile operators already exploit this tradeoff by altering the tail timer dynamically. In Figure 1 we simulate the impact of the tail timer on the energy consumption and latency for a user browsing the web and visiting a new web page every 12s (details are given in section 4).

In fact, most existing work on reducing the energy consumption of cellular links impacts latency. An active LTE link during idle periods may transition into a more efficient DRX (Discontinuous Reception) state where it alternates between listening to the channel and sleeping; this delays traffic by tens to hundreds of milliseconds but reduces energy consumption. Higher up the stack, Balakrishnan et al. [1] propose to use 3G fast dormancy to allow mobile devices to transition to idle when the likelihood of future traffic is low, inflating the latency when the prediction is wrong.
Finally, other approaches use application knowledge and explicitly delay traffic to reduce the number of costly energy tails [2]. In this paper we ask the question of how latency and energy consumption can be reduced simultaneously for cellular links. To find a solution, we leverage the key observation that cellular links are only energy efficient when idle or fully-loaded, because lightly loaded links consume just as much power as fully loaded ones. A good solution should try to move these links into one of these two states. Reduced latency will come as a side effect, because traffic will travel mostly “active” links.

Our solution targets groups of nearby devices, rather than individual ones, and it aims to consolidate cellular traffic onto as few cellular links as possible, striving to keep a few devices’ links fully utilized, and the rest of the links completely idle. In many formal or informal user communities, there are ample opportunities to share connections [3].

Our solution, the Mobile kibbutz, enables nearby mobile devices to dynamically share their cellular connections with each other via WiFi or Bluetooth links in a secure and fair manner. When a mobile device uses its cellular link for its own purposes, it also forwards traffic for its neighbors connected via local links. Measurement results show that the added cost of forwarding and of the local links is significantly smaller than the cost of enabling the cellular link of all the devices, so the aggregate energy consumption is smaller.

There have been many previous works proposing connection sharing, including [4, 5, 6, 7, 8, 9], and mobile operating systems are enabled by default with the ability to “tether” traffic. Most of these works focus on bandwidth or minutes sharing, and it is enabled by the user and static. The kibbutz also uses tethering, and its novelty is in addressing the key problems related to reducing energy consumption on mobile phones via collaboration:

- Is application agnostic, and the first work that breaks the aforementioned power-RTT tradeoff.
- Provides an algorithm that allows dynamic and continuous changing of roles (multiple times per minute), to ensure that all the devices gain from collaborating. Can shift traffic onto different paths seamlessly by using an optimized version of Multipath TCPs.
- Includes accounting mechanisms that ensure users are only billed for their own traffic, and a tit-for-tat mechanism that ensures fairness in energy consumption.

We have implemented and tested the kibbutz both in simulation and on Android devices. Our experiments show that the kibbutz is very robust, fair, and offers benefits for a wide range of traffic patterns. Android tests shows that unmodified applications can benefit greatly from the kibbutz: two devices listening to (different) online radio streams with the kibbutz leads to 25% less energy consumption; typical web searches are 1.2x faster on average, and 4x faster in the 90’th percentile; downloading mobile apps takes 30% shorter.

2. MOBILE NETWORKING TODAY

We begin by discussing the basic properties of radio technologies available on mobile devices today (cellular, WiFi and Bluetooth), attempting to understand how they cater for the traffic patterns of common applications.

Cellular connectivity such as 3G and LTE is widely available geographically, being the default choice for devices. Cellular power consumption depends on the state of the connected link: when the link is idle, its power consumption is close to zero. When the device has to transmit or receive a single typically-sized packet (e.g. 1KB), it will transition into a high power state. The transition takes seconds and is expensive energetically because of signaling with the base station. To amortize its cost across many packets, the solution used in practice is for the mobile device to remain in the high-power state for a predefined amount of time after the last packet has been sent (5-10 seconds, depending on the operator). When very little traffic is sent, this tail wastes precious energy, draining batteries. This behavior is widely studied in recent literature [2, 10, 1, 11], and also visible in our own experiments (see Figure 9 left, labeled ‘Stand alone’). Any relevant transfers are performed in the high power state (called DCH in 3G, and CONNECTED in LTE).

In our 3G tests, the transition to DCH state takes about 2.3s up, and 7.3s down for one operator. The sum of these values seems to vary slightly with time of the day, but stays roughly in the 5s-10s region. This behavior has two critical implications for interactive traffic. First, after a silence, a mobile using 3G will have to wait the 2.3s setup time before sending any traffic, and second, waste about 3s of energy to wind down. In addition, any amount of traffic with a period of less than 5-7s will keep the radio in the high power state, regardless of the actual transfer rate. In fact, 3G power consumption tends to have a flat response to increasing traffic. Beyond a certain inter-arrival rate, the 3G interface remains in the high power state, and variations in consumption depend only on the cost of moving data through the stack, as confirmed by measurements in Figure 2(b).

LTE networks behave similarly (Figure 2(a)). There are quantitative differences, though: total power consumption is higher, on the order of 1.0-2.5W, while the setup time is a bit smaller, between 0.2s-0.5s for the MNO (Mobile Network Operator) we have tested. We found that DRX is used both in RRC (Radio Resource Con-
nected), and in IDLE states, with a 1ms short timer. Other authors report similar findings [11, 1].

WiFi links offer better throughput than their cellular counterparts, but their coverage is patchy. Most phones use the 802.11 power save mode permanently in order to power down the interface when little traffic is to be transferred, making WiFi power consumption proportional to the traffic rate. In Figure 2(c), we plot the power consumption for different rates, and it is mostly linear for rates above 200Kbps. Part of the linearity of the curve is given by the cost of the software stack, as was recently shown in [12].

Bluetooth links (3.0 in our phones) are only available at a few meters and provide 2Mbps in our tests. On the other hand, Bluetooth power consumption curve is linear with the throughput achieved (Figure 2(d)).

To emphasize the difference in power consumption patterns between the four radios, in Fig. 2(e) we recast the measured data to show the efficiency measured in Mbit/J. The efficiency does increase for all three technologies due to amortization. For 3G the tail end gets used better with increasing traffic, when more bits are handled with the same cost. For Bluetooth and WiFi, powering the circuitry takes proportionally more than the software stack - in Figures 2(c) and 2(d) the line doesn’t actually pass through 0. Note that at lower rates, the efficiency of 3G is much worse, due to the long inactivity timer in the DCH and FACH states. Beyond 750Kbps WiFi is more efficient than Bluetooth.

Mobile applications fall in two broad classes. Background apps synchronize periodically with servers, have low rates and waste energy because of the tail timer. Optimizing energy consumption for such apps has been an active research area, with quite a few solutions proposed [13, 14, 2, 10, 1].

Interactive applications can be loosely characterized in other three classes: streaming (VOIP, radio, online gaming, video streaming), web browsing and app downloads.

Streaming apps utilize cellular links inefficiently; they consume little bandwidth (tens to hundreds of Kbps) and have large packet inter-arrival times. Web browsing works in a user-driven closed-loop: a burst of traffic downloading a page triggers a costly energy tail, followed by a relatively short period of silence that allows the radio to go idle. When the next burst arrives, it has to wait for a state transition. Finally, app downloads utilize the full speed of the cellular interface for brief periods. Here, higher download speeds are preferable, but users are limited by the cellular capacity.

Running these apps over cellular links leads to high energy costs and poor overall performance. Ideally, we would like to run all these applications over WiFi-like links that offer energy proportionality and high speeds, but such links are not available all the time.

3. THE MOBILE KIBBUTZ

The key insights from our measurements are that:

• Cellular links offer good interactive experience only when in the high-power state. In this state they draw (roughly) the same power regardless of the traffic load.

• Many mobile applications such as web radio, or web browsing under-utilize cellular links most of the time.

• WiFi and Bluetooth links offer power-proportionality and low delays.

Our goal is to achieve low energy consumption, low RTTs, and fair privacy aware sharing of cellular links, all while maintaining the TCP socket interface to existing applications. To the best of our knowledge, there are no solutions that achieve this goal today.

All existing work on optimizing energy consumption for cellular links trades latency for energy efficiency by adaptively reducing the tail timer or clustering packets in time by delaying some of them. The first solution works reasonably well for background applications, but it is not applicable to applications that send at low data rates (e.g. radio, streaming). The second solution requires application changes, and only applies to delay-tolerant apps. Even PHY-layer solutions such as LTE’s DRX use the same tradeoff, decreasing energy consumption for increased delay.

Our solution is based on the observation that, in most cases, mobile users roam in dense areas where many other mobile users are around (public spaces, transport, office, home). We propose to join nearby mobile devices into a Mobile kibbutz, which allows devices to use each other’s cellular connection by forwarding traffic along cheaper WiFi or Bluetooth links. By joining the kibbutz, devices can dynamically share their cellular connections with the rest of their group for an advertised period. While the advertisement is valid, neighboring devices have a choice between using their own link or the shared link for sending or receiving traffic.

The kibbutz allows devices to consolidate their traffic onto one or a few cellular links, reducing both energy consumption and delay across all devices. Consolidation brings cellular links closer to the their optimal operating point: one link will be highly loaded while others will be idle. Intuitively, the kibbutz saves energy because: a) local links are power proportional and b) the added energy cost of forwarding data on behalf of others is very small compared to the cost of using the cellular link.

The concept is presented in Figure 3. The black user is listening to Internet radio, while the red user is browsing the web. In isolation, both users keep their cellular links busy most of the time (Fig. 3.a), despite the low data rate. In addition, the red user experiences high delays when accessing every single page because his link must transition from idle to the connected state. By consolidating traffic onto a single link we can leave the other link IDLE all the time; this reduces the total energy consumption to approximately half. Additionally, the red user’s delay decreases because state transitions are unnecessary most of the time.

Users may leave a kibbutz when mobile: we must ensure that connections are not disrupted in such cases. To this end, the mobile devices use MPTCP to dynamically steer traffic over one or a subset of available links. In effect, the kibbutz pools together cellular links, enabling many optimization avenues. In the rest of this section we describe in detail the operation of the mobile kibbutz.

3.1 Basic Operation

To use the mobile kibbutz, a device will automatically connect to other nearby devices. There are a number of technologies that can be used for this purpose:
• Bluetooth is the most energy efficient but it only supports point-to-point connections, so it will be difficult to create large groups of devices because of the sheer number of connections required. Additionally, Bluetooth has relatively low speeds (2Mbsp in our tests).

• WiFi hotspot mode (or WiFi direct) is when one device acts as the access point, and the other as client. This mode is widely supported and offers high speeds. On the downside the setup is asymmetric as the access point will consume more power (200mW on the Galaxy Nexus).

• WiFi Ad-Hoc is our favorite technology because it allows true peer-to-peer operation and high speeds; unfortunately support for Ad-Hoc is being phased out of mobile WiFi drivers.

• LTE Direct is an emerging technology that uses the LTE band for energy-efficient device-to-device communication and discovery. LTE Direct appears very promising for our purposes but is not yet available.

The mechanisms needed to automatically join a kibbutz depend on the local link. WiFi ad-hoc is easiest: a device willing to share a link will send broadcast messages that prospective clients can hear and respond to. For WiFi hotspot mode, a device willing to share its cellular link will become an access point using some predefined SSID. To join the kibbutz, prospective clients will simply associate to this AP. Finally, BT devices will set their device names to some predefined values, and devices willingness to enter the kibbutz will search for other devices and allow themselves to be discovered; unsafe pairing can be done automatically, without requiring user input.

Advertisements. Once the device is connected to a kibbutz, it can share its cellular connection with other devices. One simple option is to always share the connection, but this may quickly drain the device’s battery. A better option is to share the link only when it is already in the high power state: this way the additional energy consumed to forward the neighbor’s traffic is very small and the traffic will experience low delays. Device A will advertise its link when one of its own packets leaves the interface and the interface is not already shared. A’s advertisement includes an expiration time t that tells peers when to stop forwarding traffic via A.

The choice of t affects the performance of the kibbutz: smaller t leads to more signaling required for advertisements, and it could also lead to frequent switches of traffic between devices, thus negating the energy savings the kibbutz can bring. That is why we use an advertisement period equal to the tail timer of the mobile operator – the largest time that guarantees the link will always be in the high power state when used by peers. Sharing an IDLE link would be bad: it costs a state transition and provides increased delay.

The default advertisement algorithm can lead to unfairness: in many cases the whole kibbutz will end up using a single device’s link. Additionally, it is very easy for a device to leech on other devices’ resources, undermining the kibbutz. To incentivize devices to share their links we adopt a simple tit-for-tat algorithm: each device keeps a counter for each neighbor indicating the number of times it will allow the neighbor to use its link without the neighbor reciprocating. The counter is decreased when the neighbor first uses the link during an advertisement period, and increased when the device uses one of the neighbor’s advertisements. When the counter reaches zero, the link is no longer advertised.

The initial value of the counter must be positive and is a parameter of the algorithm. Larger values are more energy efficient as they reduce the need to switch between links; at the same time they can create short-term unfairness in link usage. We explore this tradeoff in simulation in Section 4.

Traffic Optimization. Each device keeps a list of received advertisements that it can use to optimize different goals as follows:

• Save Energy: devices will always send traffic via advertised links; they will only use their own link when no other links are available. This is the default strategy.

• Maximize Bandwidth: devices will use all advertised links as well as their own in a bid to maximize their throughput.

• Minimize RTT: the Save Energy strategy will also reduce the RTT because advertised links are by design in the high power state which gives small round-trip times. However, the horizontal link used to reach the neighbor also adds precious milliseconds to the round-trip time; devices wanting the absolute smallest RTT might prefer their own link in the DCH state to advertised links.

When device A wishes to use neighbor B’s link it will simply route packets via the “local” link to B. B will receive packets on the local link, change their source address and port (i.e. NAT) and place them onto its cellular link. Return traffic reaching B will go through the (reverse) NAT and then will be forwarded to A. In effect B is acting like a mobile hotspot for A - and this is already supported out of the box by most mobile platforms.

New TCP connections will work seamlessly over the new link, without the applications needing to know which link their traffic is taking. As long as these connections finish before the advertisement period ends, everything is great.

However, connections that last longer than the advertisement period will simply be terminated, as they are bound to the IP address of the advertised cellular link. This is a fundamental limitation of TCP: once a connection is created, it is bound to the addresses of the endpoints. Change the addresses and the connection will just die. As 97% mobile traffic runs over TCP [16] and many connections are long-lived, this limitation severely restricts the utility of the kibbutz.

Multipath TCP. To take full advantage of the kibbutz we use the newly standardized MPTCP protocol [17]. It is an evolution of TCP that allows unmodified applications to use multiple paths in the network through a single transport connection. MPTCP connections are composed of one or many subflows that appear like a single TCP socket to the applications. By default, MPTCP will use all available subflows to send data, favoring connections that have lower loss rates [18]. With MPTCP, endpoints can add and remove subflows at any point during the lifetime of a connection. Using this feature we can redirect any connection to use the cellular or neighboring links by adding new subflows. To force the connection’s packets via the new subflow there are two options: we can close the cellular subflow, forcing the remote end to use the new subflow, or announce the remote-end that the direct subflow has lower priority. We use the second option because it avoids repeated setup and tear-down of subflows over the device’s cellular link. By default, all direct subflows are marked as low priority; subflows routed via neighbors are marked differently depending on the optimization goal.

In the energy saving profile, neighbor subflows are marked as normal priority, thus all traffic will prefer them to own 3G links; these will only be used when all the neighbor links experience repeated timeouts.

In the bandwidth and RTT profiles all subflows will be marked low priority, and transition between the two modes is done auto-
authenticate the mobile device via the WiFi hotspot which relays this information to the relevant MNO servers. The kibbutz partner acts as an offloading point from the point of view of the mobile, as the EAP conversation is performed over the partner’s own 3G link. Once the authentication process is completed, the kibbutz partner knows whether to allow the traffic through or not. The key resulting from the authentication exchange is then used by the mobile device to establish an IPsec tunnel that is terminated in the operator’s network. All other policy and charging functions are reused from the existing core. Note that MNOs already have all the equipment required to support mobile data offloading, and mobile devices support the required technologies out of the box. The only downside of such offloading is that encryption is done in software.

One alternative that avoids the overhead of software-based encryption is proposed by the Swedish startup AnyFi [21]. It allows a WiFi device authenticate against its home network even when away from home, by tunneling the authentication, association, and encrypted data through local access, through WiFi in IP encapsulation. This does not require any changes on the mobiles, but requires AnyFi support on the home AP(s) (a back-end) and foreign AP(s) (a front-end). AnyFi is being considered by many MNOs \(^1\) for mobile data offloading as it has the advantage of using the hardware AES present in any WiFi card, and it simplifies authentication for both the user and the network. Adapting this architecture for the kibbutz entails only running an AnyFi front-end on the phone that accepts to encapsulate and route traffic from other kibbutz peers on its 3G/4G uplink. AnyFi can also be easily extended to tunnel Bluetooth instead, thus enabling secure Bluetooth links.

4. SIMULATION ANALYSIS

In this section we use simulation to understand the basic properties of the kibbutz across parameters that cannot be easily controlled in practice. We show experimentally in Section 6 that these benefits can be obtained on real devices.

We have implemented a discrete-time simulator that services scheduled traffic requests from devices. To model 3G power consumption the simulator tracks whether a device is in the high power state, and the tail timer is given as input. Cellular link speed is set to 2Mbps. RTT is measured for each traffic request; it is set to 2s if the device must transition to the high power state, otherwise it is set to 200ms\(^2\). Finally, the simulator assumes the local links are perfect, having zero delay and infinite bandwidth (BT and WiFi are mostly negligible compared with 3G). The simulator also implements the tit-for-tat algorithm and sets initial counter to five slots.

We have modeled three application classes: streaming models audio and video streaming apps that send traffic every second thus keeping the cellular link up all the time; web browsing is modeled using randomized inter-arrival time with an average of 13s (value chosen by profiling our own browsing habits on mobiles); background traffic such as podcasts, or email and was similarly modeled with an average of 30s.

Energy Savings. To understand the possible benefits of the kibbutz, we varied the number of devices and ran the three app types described above. In Figure 4(a) we show the expected energy savings from a kibbutz; the numbers given represent the percentage of time the 3G interface stays in the high power state. The best we can hope for is the power to decrease inversely proportional to the number of devices.

\(^1\) private conversations with MNO engineers

\(^2\)RTT for our 3G MNO varies, sometimes wildly, between 85ms and 900ms depending on time of day, user speed, and cell population.
Streaming keeps the radio up all the time if a single device is playing; if two devices are playing in a kibbutz, they will each save around 45% of the 3G energy. As more devices join the kibbutz, the benefits increase as expected. A similar curve is seen for web traffic; here the baseline is at around 80%, because the radio manages to sleep a little in between two pages. We observe that for these two applications even small numbers of devices make a big difference to power consumption. The story is different for background traffic; here the baseline consumption is only 30%, and it drops more slowly; with three devices it stays at 24%.

How many devices is it feasible to have in a kibbutz? Assuming the device density is high enough, the local links will become bottlenecks when many devices join. We have run experiments with two Android devices without any problems. Going to four-five devices is certainly feasible for Bluetooth links as long as aggregate traffic rate is low. Beyond that other costs will start dominating and will produce diminishing returns. With this in mind, it becomes apparent that the kibbutz does not bring many benefits for background-style traffic: the messages are so sparse in time that consolidation does not help much, especially for small numbers of devices. It makes sense, then, to only enable the kibbutz when the device is active.

RTT Reduction. 3G state transitions induce high delays to packets; our measurements show this delay to be around 2s depending on time of day, for a number of operators. Once the radio is in the high power state, delays are much lower - typically 100ms-200ms. We measured the average RTT experienced by each transfer for each app class, with the results shown in Figure 4(b). The simulated RTT decreases five times for web transfers with three devices in a kibbutz. As web transfer is RTT-bound, this reduction is significant and will enhance user experience. Background traffic is less sensitive to RTT, hence the decrease in RTT while sizeable it may not matter that much in practice. Finally, RTT increases a little for streaming apps because the kibbutz switches between different links that will have to transition to the high power state. With a single device this is not needed, as the link is up all the time.

Reduced Signaling Costs. Our previous measurements focus on user-centric performance indicators. We also wish to understand how the kibbutz affects the network, in particular reducing the signaling costs between the devices and the cellular base-station needed for state promotions. Signaling is an important optimization target for carriers [13]. To this end, in Figure 4(c), we measured the frequency of transitions to the high power state induced by our three categories of apps. Web and background traffic induce a transition every time a packet is sent when a device is running standalone. In a three device kibbutz the number of transitions is reduced tenfold for web-traffic, and halved for background traffic. Streaming will incur a bit more state transitions with the kibbutz, because instead of staying always up, the 3G link now toggles up and down whenever the mobile consolidates its traffic with the neighbors.

Effect of the Tail Timer. We wish to understand what effect the tail timer has on the key metrics for users (power) and the network (signaling). The tail timer is the only knob available to mobile operators today. To measure its impact, we vary the tail timer from 1 to 30s and measure the web application when running standalone and in groups of two or three devices.

The tail timer offers a trade-off between power (Fig. 5(a)), signaling (Fig. 5(c)), and RTT (Fig. 5(b)). The graphs show how difficult it is to choose a good trade-off today: one has to go for low signaling and delay or low power, but not both. The kibbutz changes the trade-off space: it tames power consumption for higher values of the tail timer, while also reducing RTT.

If we compare the tail timer to the average inter-arrival time between web requests (13s) we get other insights from these graphs. When the tail timer is very low, web traffic rarely benefits from the kibbutz because the radio is in the low power state most of the time. As the tail timer increases, the cellular link is up more and this allows the kibbutz to consolidate traffic and obtain benefits for both power and RTT. As the tail timer goes beyond 13s, the kibbutz only improves power usage.

Dynamic Adjustment of the Tail Timer. There is a wealth of existing research that exploits fast dormancy, a feature standardized by 3GPP [22] that allows the mobile device to demand a transition to the IDLE state, thus reducing its tail time and its associated energy cost. Using fast-dormancy can save energy but it requires that the device knows the traffic pattern of the application [13, 14, 1]. A concrete example is the MakeIdle algorithm proposed in [1] that uses machine learning to infer when it is appropriate to transition to sleep.

To understand how the kibbutz compares to these works, we have implemented an oracle that knows the traffic workload and decides after each packet sent whether to transition to sleep or not. The oracle goes to sleep only if the next packet is due after a threshold old number of seconds. The threshold takes into account the time needed for the two state transitions, each of which takes 1-2s in our measurements; we set it to 4 or more seconds.

We ran all the simulations shown in Figures 4 with the oracle algorithm and varying thresholds. We found that the algorithm has no impact whatsoever on the streaming app, and the performance is the same as on a standard mobile device. This is because streaming packets have low inter-arrival times and there is no time to sleep. In comparison, a three-device kibbutz cuts energy consumption to a third.

The web traffic results are more interesting. With a 4s threshold, the Oracle reduces the time spent in the high energy state from 72% to 17%, but the energy reduction comes at a cost: the average
gains are similar, but the well for background applications—hence the two techniques could for streaming like applications, while tail-reduction works really nevertheless, this comparative analysis shows that the kibbutz gains are modest: a three-mobile kibbutz reduces energy to 22%.

Tail reduction has the biggest positive effect on background apps, decreasing the time spent in high power state from 33% to around 8%. For such apps the kibbutz gains are modest: a three-mobile kibbutz reducing energy to 22%.

The oracle algorithm provides an upper bound for tail-reduction algorithms, and the actual benefits will be smaller in practice. Nevertheless, this comparative analysis shows that the kibbutz excels for streaming like applications, while tail-reduction works really well for background applications—hence the two techniques could complement each other nicely. For web-style traffic, the energy gains are similar, but the kibbutz also reduces the RTT and signaling costs.

Fairness. All the experiments shown so far were run for a long period (2 hours); this ensured that all the performance measurements were nearly identical across all devices.

The tit-for-tat counter governs how many times a device will share its connection without the neighbor sharing back. To understand the effect of this parameter we ran the streaming application for a short period of time (5 mins) with different values of the counter. The results in Fig. 6 show that using small values of the counter leads to increased energy consumption because of frequent switching between power states and the tail timer. Higher values of the counter are more efficient, but lead to unfairness: a single device may bear the whole cost of cellular connectivity for high values. Values above five give diminishing returns in average consumption, but increase unfairness. This is the value we used in all our other experiments.

RTT increases from 1.46s to 1.9s as the number of state transitions increases by a third. This should be avoided as it affects the scalability of 3G networks [13, 23]. In fact, carriers are afraid to enable fast-dormancy because of increased signaling traffic[23].

To reduce the effects of signaling, we increase the Oracle algorithm’s switching threshold to be the same as the tail timer (10s); this guarantees that the number of state transitions (and the RTT) will not increase. In this case, the RTT and signaling costs stay the same, while the time spent in DCH drops from 72% to 36%. In comparison, a three-device kibbutz has similar energy behavior (35% in DCH) but it reduces the RTT to 0.4s and the signaling costs by 50%.

Tail reduction has the biggest positive effect on background apps, decreasing the time spent in high power state from 33% to around 8%. For such apps the kibbutz gains are modest: a three-mobile kibbutz reducing energy to 22%.

The oracle algorithm provides an upper bound for tail-reduction algorithms, and the actual benefits will be smaller in practice. Nevertheless, this comparative analysis shows that the kibbutz excels for streaming like applications, while tail-reduction works really well for background applications—hence the two techniques could complement each other nicely. For web-style traffic, the energy gains are similar, but the kibbutz also reduces the RTT and signaling costs.

Interactions Between Different Application Classes. We now look at experiments involving different classes of applications, with results given below. A row shows an application class running in a kibbutz of two devices with another application class (the column), or in a kibbutz of three devices. Each cell lists the amount of time the 3G interface stays up; the baseline is also given on each row. For instance, the Background power consumption drops from 30% to 21% when running in a kibbutz with a Web application.

The numbers show that the application mix does influence the results but not dramatically. Also, the savings are shared between the different apps. As expected, the Background class of applications improves the least.

<table>
<thead>
<tr>
<th>App. Class</th>
<th>Back.</th>
<th>Web</th>
<th>Stream</th>
<th>All</th>
</tr>
</thead>
<tbody>
<tr>
<td>Background (30%)</td>
<td>25%</td>
<td>21%</td>
<td>21%</td>
<td>17%</td>
</tr>
<tr>
<td>Web (80%)</td>
<td>68%</td>
<td>49%</td>
<td>48%</td>
<td>41%</td>
</tr>
<tr>
<td>Streaming (100%)</td>
<td>78%</td>
<td>58%</td>
<td>54%</td>
<td>50%</td>
</tr>
</tbody>
</table>

5. IMPLEMENTATION

We have implemented the kibbutz using Bluetooth and WiFi as local links. First we enabled the devices to seamlessly route traffic for neighbors with pre-existing Android mechanisms: we created BNEP interfaces using pand(1) from the BlueZ1 suite available on Android 4.1.2 and we employed normal netfilter mechanisms to route traffic and perform Network Address Translation.

The traffic consolidation scheme and the short-term fairness are implemented in a kernel module. We use netfilter hooks to efficiently handle the traffic on every interface without polling. This way, our module knows when the mobile link is up, when we are using a neighbor’s link or when a neighbor is using our own link.

The main part of the service consists in “moving” traffic from one link to another, without changing the application. This is achieved using standard MPTCP mechanisms. Upon system startup we set the cellular link to low priority and disable MPTCP for local links. This ensures MPTCP will try to avoid the 3G link when emitting traffic and will never attempt to create flows on Bluetooth links by itself – by default MPTCP creates flows on all interfaces. The module exposes an interface permitting userspace program to declare the local links as “usable”. Once a local link (Bluetooth or WiFi) is declared usable MPTCP will consider it. Since the cellular link has low priority, any local links will win the flow selection process, i.e. traffic will move from the mobile link to the local links. Implementing the bandwidth maximizing / RTT minimizing profile is easily achieved by removing priorities and using the default scheduling algorithm of MPTCP. For the energy saving profile, we implemented the bandwidth watchdog by changing the MPTCP scheduling algorithm.

We implemented the upper part of the kibbutz as a userspace driver that uses the kernel module and controls the entire process.

1http://www.bluez.org/
This driver monitors the local link and the kernel module for advertisements. The kernel module sends up notifications like ‘when did A last use our link’, ‘when did we last use A’s link’, allowing the enforcement of short-term fairness via a tit-for-tat mechanism, as discussed in §3.

6. EXPERIMENTAL EVALUATION

We have deployed the mobile kibbutz on two Galaxy Nexus mobile devices running Android 4.1.2 (Jelly Bean) and tested it with synthetic traffic as well as representative applications. We measure power using the Monsoon power monitor. We ran most experiments using the only cellular operator in our area that allowed MPTCP (it did not remove MPTCP options from packets). This operator offered 3G (HSDPA) at advertised speeds up to 7.2Mbps.

To enable the kibbutz, we deployed a Linux box running MPTCP that we used in all experiments. The setup is shown in Fig. 7. Ideally, this box would act like a SOCKS proxy terminating MPTCP connections and opening TCP connections to the real server (Fig. 7.a). For this to work we would need proxy support from Android; unfortunately this is only available over WiFi connections, not 3G ones – this seems to be a policy option rather than a technical one. A work-around is to open an MPTCP-based tunnel to the proxy, and tunnel all device traffic (Fig. 7.b). In this case, the MPTCP-based tunnel is encrypted. This setup works fine, however tunneling TCP connections over (MP)TCP can lead to bad performance interactions that could bias our experiments. That is why we used the third setup (Fig. 7.c) for most of our experiments: here our box also serves content, running stock HTTP and streaming servers.

In most of our experiments we used 3G with Bluetooth as the local link. We prefer Bluetooth as it has has low, symmetric energy consumption across connected devices; RTT is 7ms, less than 9% of the minimum RTT obtained for our 3G MNO. We also tested the kibbutz with WiFi connectivity, having one device act as an access point and the other as client. This setup offers better performance at the cost of increased energy consumption; it was used for our download tests. Finally, we tested over LTE cellular link with both Bluetooth and WiFi as local links to understand how applicable the kibbutz is.

Below we describe our practical findings. In brief, they confirm our simulation results showing that the kibbutz significantly reduces the power costs associated to using the cellular links and their RTTs. We also find that the network represents a large fraction of device consumption: using the mobile kibbutz with 3G and Bluetooth reduces total phone energy consumption by 25% for radio and web browsing.

6.1 Basic Functionality

To test the basic operation of the kibbutz, we ran experiments where each device runs a simple client that opens a long-running TCP connection and then sends a timestamp twice per second to the server which just echoes it back. The clients measure the app-level RTT and print it.

In Figure 8, the upper curve shows the power consumed by Device1 as it switches its 3G card on and off. The segments at the bottom correspond to the packets received at the server from Device1 or Device2. The upper segment sequence corresponds exactly with the power consumption recorded by the power-meter, confirming the fair behavior of the kibbutz. If a device uses only its link, it needs 754mW on average, whereas it only needs 169mW if it uses its neighbor - the upper and lower levels of the power measurement. The average power consumption of Device1 is 480mW. This is a saving of 37% over its maximum consumption if it used only its own 3G connection.

We also tested power consumption with periodic bidirectional traffic of 24Kbps (e.g. VOIP) for all combinations of hardware of interest for the kibbutz. In the following table we summarize power measurements in mW for these link combinations: 3G and 4G for the vertical link, and BT or WiFi for the horizontal link. The kibbutz server uses its vertical link to route packets from the kibbutz client obtained on the horizontal link. In all WiFi experiments we use WPA2, therefore including the cost of traffic encryption (BT is encrypted by default).

<table>
<thead>
<tr>
<th>Link type</th>
<th>Router [mW]</th>
<th>Consumer [mW]</th>
<th>Standalone [mW]</th>
<th>Power Savings</th>
</tr>
</thead>
<tbody>
<tr>
<td>LTE + BT</td>
<td>1075</td>
<td>220</td>
<td>1050</td>
<td>35.3%</td>
</tr>
<tr>
<td>LTE + WiFi</td>
<td>1316 / 1080</td>
<td>170 / 320</td>
<td>1020</td>
<td>27.1% / 31.4%</td>
</tr>
<tr>
<td>3G + BT</td>
<td>740</td>
<td>191</td>
<td>675</td>
<td>31.0%</td>
</tr>
<tr>
<td>3G + WiFi</td>
<td>866 / 630</td>
<td>113 / 323</td>
<td>680</td>
<td>28.0% / 29.9%</td>
</tr>
</tbody>
</table>

When using WiFi as a local link, the energy consumption is higher for the mobile acting as access point. That is why we give two numbers, one when the mobile is also the access point, and one when it only is a WiFi client. As expected, LTE has higher power consumption, and provides ample savings opportunities. Using BT as a local link for 3G is the best choice: using WiFi would not increase throughput by much, and would slightly increase energy consumption. For LTE, using WiFi makes sense as it allows greater speeds. Unless stated otherwise, the experiments in this paper are run over 3G and Bluetooth.

The kibbutz reduces energy consumption for all possible combinations of cellular and local links, and the savings for our hardware/MNO setups are of 27% - 35%. These relative gains are for the “best case” scenario with synthetic traffic, that is not consumed by an actual application. Next we show significant benefits can be had even when real apps are running.

6.2 Improving Popular Applications

We four typical applications that depend on network connectivity: audio streaming (e.g. radio, VOIP), video streaming, web browsing and app downloads. While not exhaustive, we believe these capture the main “active” traffic patterns on mobile devices. We consciously focus on active apps, leaving out background apps such as email that generate traffic rarely. We do not expect worthwhile improvements for such apps, especially when we consider the overheads for running the kibbutz (measured in Sec. 6.4).

Streaming apps. We installed Apache on our server and VLC player on the mobile phones. For audio streaming we served MP3 files encoded at 48Kbps. The Galaxy Nexus phones consumed 976mW when streaming radio alone, with the screen off. Most of this cost is due to the 3G link. When running a two-device kibbutz with both streaming the same file, instantaneous power consumption was bimodal: the device routing data consumed 1047mW (slightly more than standalone), while the other device consumed only 403mW. The two devices traded roles periodically.

The average power consumption for a device in the kibbutz was 723mW, 25% less than standalone consumption. A device streaming radio in a kibbutz will last 34% longer than one running standalone. The savings in absolute terms—around 250mW—are a bit less than half the energy consumed by the 3G radio in the high power state (650mW) which is the maximum gain that can be had in this configuration: the kibbutz is working as expected.

For video streaming we served a YouTube clip encoded at standard definition (250Kbps). Here standalone energy consumption was 1819mW - this value includes the cost of the screen, as well as of the GPU used for rendering. Consumption was again bimodal in the kibbutz: 1890mW for the forwarding device, and 1205mW...
some images. We verified for all the sites used that the browser and
google searches include another larger transfer of 20K-50K, with
CDN server, and more small transfers occur. Most newspapers and
about halfway through, another DNS lookup is made, usually to a
or scripts get downloaded progressively as the main index is parsed;
hibit the following behavior: after a DNS lookup, a main index file
loading typical pages from several mobile optimized web sites in-

Web Browsing. Popular web pages today are optimized to the
point where their load times are determined by the RTT rather than the
bandwidth. That is why Google has been actively promoting a
number of TCP optimizations aimed at reducing the number of
RTTs required to download a webpage, including a proposal to in-
crease the initial TCP congestion window to 10 packets [24] and
to implement a flavor of transactional TCP called TCP Fast Open
[25]. On cellular links, the RTT after idle is very large, dominating
the load time. Can the kibbutz help alleviate this problem?

Measuring web traffic is tricky. We cannot simply replicate web-
pages on our server because modern sites link many objects, some of
which are dynamically generated and on different sites. That is
why we ran two types of web browsing experiments: an MPTCP-
based tunnel and emulating web-traffic. The tunneled experiment
has a few potential pitfalls: it is prone to TCP-over-TCP perfor-
ance issues, and it uses a “visual” method of measuring the load
time. The emulated web traffic experiments avoid both problems:
they give precise timings and avoid encapsulation problems, but
may not model web traffic as accurately. Both experiments showed
good improvements for the kibbutz, which gives us confidence the
benefits will appear in practice.

In the tunneled setup (shown in Fig. 7.(b) we repeatedly loaded
a Google search page using a single device over 3G, or a device in
a kibbutz using its always-available neighbor’s advertisements.
In both cases the screen is on and the device’s browser renders the
page. To measure delay we use the power meter readings, identifying
the peaks of consumption that appear during page load. These
are shown in Fig. 9. Downloading and displaying the Google
search page takes 5.6 seconds on average, and it costs around 15.9
Joules. When using a kibbutz, browsing the same page takes 3.2
seconds and costs 3.9 Joules. The kibbutz loads the search page
2.4s quicker, just a bit more than the state promotion time mea-
sured for this provider.

To emulate web traffic, we analyzed tcpdump traces obtained by
loading typical pages from several mobile optimized web sites in-
cluding BBC, CNN, Amazon, and Google search. Most sites ex-
hibit the following behavior: after a DNS lookup, a main index file
of 20K-50K is started; smaller transfers of 1K-5K of images, ads,
or scripts get downloaded progressively as the main index is parsed;
about halfway through, another DNS lookup is made, usually to a
CDN server, and more small transfers occur. Most newspapers and
google searches include another larger transfer of 20K-50K, with
some images. We verified for all the sites used that the browser and
the server use HTTP 1.1 with persistent connections, therefore one

![Figure 7: Experiment setup](image)

![Figure 8: Simple TCP ping on two devices in a kibbutz.](image)

![Figure 9: Loading a Google search results page](image)

for the other. Average consumption for devices in the kibbutz was
1547mW—15% less than standalone. A device streaming video
in a kibbutz will last 18% longer than one streaming on its own.

App Downloads. How long does it take to download an average
mobile application alone or in a kibbutz? We downloaded repeat-
edly the same 6MB file from our server (this is the average size of
Android apps [26]), measuring download time and energy con-
sumption. For a standalone device, the download time was 31 ±
8 seconds. The high variance is due to competing traffic in the HS-
DPA network we are using. The phone draws 1.3Watts during this
download, consuming on average a total of 41 Joules.

For the kibbutz, one device generates low-rate traffic using our
synthetic client. The other device is running in bandwidth opti-
mization mode, using all links it has available. This kibbutz uses
WiFi as the local link to ensure the best throughput (the bandwidth-
hungry device acted as client, and the other acted as Access Point).

HSiDPA dynamically assigns all available cell bandwidth to ac-
tive users, so joining two users connected to the same cellular bases-
tation in a kibbutz will not always increase the total bandwidth
available to them, yielding similar download times to the stan-
dalone test. If the two devices are in different cells or sectors,
bandwidth should always increase.

We confirm this hypothesis by using a different operator for our
second device - this operator strips MPTCP options, so we tunneled
MPTCP traffic over UDP. Downloading the same file in the kibbutz
takes 21 ± 5 seconds, decreasing download time by a third on
average. Counter-intuitively, the kibbutz also saves power despite
using more network interfaces: the total energy consumed to down-
load the file is 31 Joules, 25% less than standalone.

6.3 Robustness

If a mobile device leaves the area of a kibbutz, how will it affect
other kibbutz users? To answer this question we ran an experiment
where one user is downloading a large file via its neighbor. While
the transfer is taking place, the neighbor suddenly stops forwarding
packets. Figure 10(b) shows the progression of the connection se-
quence number against time, with the disruption happening around
time 14s.
Our watchdog mechanism quickly detects after one timeout that the prioritized path has poor performance, so it starts sending packets via the direct path. After another few seconds—needed for the idle 3G link to become active—traffic resumes via the direct path.

Sudden path blackouts as above will be relatively rare: it is more likely that a path will become worse as the user is moving away from the kibbutz, leading to increased packet loss rates. High packet loss rates may appear also when there are too many users sharing one link at the same time.

To understand how the kibbutz fares in such cases, we had one device download a 2MB file via its neighbor which also dropped a fraction of the packets it forwarded. Figure 10(c) shows how the total download time varies as the loss rate varies. Vanilla MPTCP always prefers the path via the neighbor (top curve), which dramatically increases download times when loss rates are high. The watchdog detects performance problems and starts using the direct path, managing to have similar performance regardless of the loss rate. We also show the fraction of bytes sent via the direct path: with no loss all bytes are transported via the neighbor. With higher loss rates, most bytes are sent using the direct link.

These experiments outline the importance of using MPTCP for load balancing across links: single path mobility solutions must commit to a single path, while MPTCP can dynamically choose the best path at any time.

6.4 Kibbutz Overheads

The kibbutz brings benefits when there is cellular traffic, but how much does it cost when the phone is idle? We measured how much an idle device running the kibbutz software consumes compared to an idle device without the kibbutz. We expect higher costs due to Bluetooth networking and receiving kibbutz advertisements.

Our Galaxy Nexus phone draws 12mW with screen and all wireless links except 3G off (3G was in the IDLE state). The same phone running the kibbutz draws 60mW - the kibbutz adds 48mW to the baseline power consumption. We broke down the power usage as follows: starting Bluetooth raises consumption to 28mW, and enabling IP connectivity over Bluetooth (Pan) further raises consumption to 50mW. Finally, receiving kibbutz advertisements increases to 60mW.

The kibbutz brings most benefits for applications that send traffic at short intervals. Background traffic benefits less from the kibbutz: a simple optimization is to run the kibbutz when the traffic rate is greater than some minimal threshold. This reduces baseline costs to almost zero, while giving the same benefits for busy applications.

Another question relates to the cost of encryption. It is best if the kibbutz leverages the local NIC’s encryption (i.e. WiFi WPA or Bluetooth) in conjunction with AnyFi at no added cost. Without AnyFi (or similar technologies), software encryption is required.

We have run measurements to understand how much this costs, and found the overheads to be relatively small. For instance, listening to radio over Bluetooth costs 403mW, and sending traffic via an encrypted tunnel adds a further 100mW. Even with encryption, kibbutz devices consume 21% less energy than standalone.

7. RELATED WORK

The importance of cellular links to the mobile experience has long been recognized, and the result is a wealth of research measuring the properties of cellular connectivity [13, 11], and proposing solutions to optimize them ([14, 2, 10] to name only a few). Most of this research has focused on reducing energy consumption of cellular links by adopting one of two high-level approaches. The first approach, discussed in our simulation analysis, includes techniques to reduce the tail timer via fast-dormancy [1, 14, 27].

The second approach uses application knowledge to optimize the 3G link usage; for instance, [28] instruments email clients to batch operations to network and flash memory. Transmissions from different applications are batched by delaying subsets of traffic to reduce the tail energy in TailEnder [2] or to use the cellular link when it is cheapest in Bartendr [10]. Stratus [8] uses a proxy in the wired domain to compress the mobile traffic and bunch it up in time, thus reducing energy consumption. Finally, Catnap [7] also uses a proxy to batch application traffic to allow the WiFi link to sleep. However, these techniques are applicable mostly to background, delay-insensitive traffic: either apps must be changed to take advantage of it, or the system must infer which traffic can be delayed safely which is error prone.

The basic idea of the kibbutz is to opportunistically use other devices’ links, and this idea is not new, with a raft of work already published including [4, 5, 8, 6, 29, 30, 31]. The mobile kibbutz allows energy optimization across many devices, and is complementary to existing works. It offers the biggest benefits for applications that send data (almost) continuously such as audio and video streaming, as well as web browsing. These applications are highly popular on mobiles, yet they are not supported by existing works. Beyond energy, the kibbutz significantly lowers the impact of state promotion delays and reduces signaling overhead - to our best knowledge, ours is the first work concurrently reducing power, RTT and signaling cost for cellular links.

PRISM [4] proposes aggregating cellular links of nearby mobiles to increase throughput and network utilization. The focus was on changes to TCP that allow splitting to happen: we rely on Multipath TCP instead which has been carefully designed to work in today’s networks [32]. Multipath TCP has been adopted by Apple in iOS7 [33] and MPTCP proxy support is already available from Citrix[34].
Combine [5] proposes aggregating cellular links across groups of mobiles with the purpose of increasing bandwidth and focuses on an accounting mechanism that creates a market to enable bandwidth sharing. CoolTether[6] aims at providing high bandwidth to laptops by leveraging mobile phones in the vicinity, while reducing the energy drain on the mobile phones, but only focuses on HTTP. It uses a gatherer in the mobile domain to “gather” all data for a webpage and then sends it in one go to the mobile device. CoolTether relies on a local mobile device community to increase aggregate download capacity and attempts to even out battery consumption across devices, however raises privacy concerns when the devices report their battery levels to the gatherer.

In contrast, the kibbutz targets the case when the mobiles themselves are traffic clients. There is no centralized entity (like the gatherer) in the kibbutz: devices take turns in sharing their cellular links, and traffic optimization is completely distributed and seamless. Furthermore, the kibbutz is applicable to unchanged applications, and reduces overall mobile energy consumption, not just tethering consumption.

Using radios selectively based on their range vs. energy trade-off was studied in the context of WiFi/BT in Coolsots [35], also without changing applications. The kibbutz extends this to 3G/4G links, and actually uses WiFi and BT as low power horizontal links for peering. Also, the use of MPTCP brings transparency, consolidation, and liquidity for the vertical 3G/4G links used.

Another recent work [36] shows that MPTCP can be tuned to improve energy consumption and performance on devices with multiple radios.

8. CONCLUSIONS

We have described the Mobile kibbutz, an opportunistic solution to reduce the energy consumption of cellular links. The kibbutz enables one key optimization: consolidating all traffic from a set of devices onto as few cellular links as possible. Consolidation increases the energy efficiency of links that are in active use, allowing other links to sleep thus saving energy. Active links will be “always-on”, providing low round-trip times and fewer state transitions.

We have implemented the kibbutz and tested it on Samsung Galaxy Nexus devices: a two-device kibbutz; nearly halves the networking related power consumption. In particular, two Galaxy Nexus devices can stream radio 35% longer than if they were running independently. Further, web browsing experiments have shown that the median page load time is 1.2s faster with the kibbutz.

Until now, the only knob available to optimize cellular networks has been tweaking the inactivity timers; these allow trading device energy consumption for delay and signaling. The kibbutz adds a new optimization dimension that can be exploited: traffic consolidation. We have shown that on its own, consolidation can reduce energy consumption, RTT and MNO signaling at the same time. In the future, it would be interesting to see how a network can be optimized by using these knobs simultaneously.

Acknowledgements

This work was partly funded by Trilogy 2, a research project funded by the European Commission in its Seventh Framework program (FP7 317756), and by Mobi4, a research project funded by the Romanian Government (UEFISCDI 11/2012).

9. REFERENCES


