Coordinating Cellular Background Transfers using LoadSense

Abhijnan Chakraborty, Vishnu Navda,
Venkata N. Padmanabhan, Ramachandran Ramjee
Microsoft Research India

ABSTRACT

To minimize battery drain due to background communication in cellular-connected devices such as smartphones, the duration for which the cellular radio is kept active should be minimized. This, in turn, calls for scheduling the background communication so as to maximize the throughput. It has been recognized in prior work that a key determinant of throughput is the wireless link quality. However, as we show here, another key factor is the load in the cell, arising from the communication of other nodes. Unlike link quality, the only way, thus far, for a cellular client to obtain a measure of load has been to perform active probing, which defeats the goal of minimizing the active duration of the radio.

In this paper, we address the above dilemma by making the following contributions. First, we show experimentally that to obtain good throughput, considering link quality alone is insufficient, and that cellular load must also be factored in. Second, we present a novel technique called *LoadSense* for a cellular client to obtain a measure of the cellular load, locally and passively, that allows the client to determine the ideal times for communication when available throughput to the client is likely to be high. Finally, we present the *Peek-n-Sneak* protocol, which enables a cellular client to "peek" into the channel and "sneak" in with its background communication when the conditions are suitable. When multiple clients in a cell perform Peen-n-Sneak, it enables them to coordinate their communications, implicitly and in an entirely distributed manner, akin to CSMA in wireless LANs, helping improve throughput (and reduce energy drain) for all. Our experimental evaluation shows overall device energy savings of 20-60% even when Peek-n-Sneak is deployed incrementally.

Categories and Subject Descriptors

C.2.1 [**Computer Communications Networks**]: Network Architecture and Design—*Wireless communication*; C.4 [**Performance of Systems**]: Design Studies, Performance Attributes

General Terms

Algorithms, Design, Measurement, Experimentation, Performance

Keywords

Cellular, Load, 3G, LTE, 4G, Energy Saving, Background Transfers

1. INTRODUCTION

Background communication on cellular devices such as smartphones has been growing in importance [5], due to applications such as social networking and software updates. Users want gratification (e.g., the Facebook photos posted by a friend) sooner rather than later, and are typically unwilling to accept a long delay, e.g., several hours. Also, despite its prevalence, WiFi is often not usable, because of sign-in or payment requirements, vehicular mobility, etc. So the cellular link becomes the only means of communication in many situation.

However, the battery drain due to background cellular communication has been a matter of concern. Indeed, there has been much research on quantifying the energy costs, both due to transmission and reception [15] and signaling-induced radio-tail periods [14]. For these reasons, many mobile platforms (e.g., Apple iOS, Microsoft Windows Phone) restrict background communication.

Battery drain due to communication is lower when the throughput is higher, because the cellular radio remains in the active state (DCH state in 3G parlance) for a shorter duration. A key determinant of throughput is the link quality, which is quantified by the strength of the base station's pilot signal, as recorded at the receiver. Prior work on energy-efficient scheduling [15] has suggested preferentially communicating while the link quality is good.

However, link quality does not yield the full picture. *As our first contribution, we show experimentally that cellular load also matters.* The load is caused by the communication of other clients in the cell and also interference due to the neighboring cells (the latter in cell-edge locations), and fluctuates over time scales of several seconds, even when a client is stationary. The higher the load, the more the channel resources get divided, and so the lower the throughput obtained by any one client.

In view of the above, we would ideally want to schedule background communication by factoring in the load as well as the link quality. However, unlike the link quality, which can be measured passively by a cellular client, estimating load has hitherto required active measurements such as a trial download or packet-pair probes. This is because the closed nature of the cellular network does not allow a client to listen in on the communication of other clients, and the network does not share load information with the clients. However, active measurements defeat the goal of saving energy since even a single-packet probe would cause the radio to remain in a high-energy state for several seconds, even with optimization such as fast dormancy [14].

Our second contribution is a novel technique called LoadSense that allows a client to obtain a measure of cellular load, locally and passively, i.e., while keeping the radio in the idle state. LoadSense is based on sensing the total power in the cellular channel and comparing it with the power of the pilot signal transmitted by the base station. We compute the ratio of the pilot power to the total power as the power ratio. When the cellular load is nil, the total power would equal the pilot power, so the power ratio would be 0dB. However, in the presence of load due to other clients, the total power would be greater than the pilot power, since the measurement interval would span both the pilot phase and the subsequent communication with other clients in the cell. So the power ratio would be lower, as low as -10dB when the full power of a fully-loaded channel is used, assuming typical allocation of 10% of total power to the pilot channel. In practice, the power ratio could be even lower because of inter-cell interference at cell-edge locations.

We show experimentally that, while extreme values of link quality and the power ratio can, individually, help predict low or high throughput, there is a wide range in the middle where neither metric, by itself, enables accurate prediction. Therefore, LoadSense uses a support vector machine generated classifier that works with a set of readily-available features, including link quality, power ratio, and the observed variation in the power ratio, to make a binary prediction of the availability of low or high throughput for the cellular client. We show that our classifier is efficient to compute on the client, utilizes only three seconds of passive measurements to make its prediction and achieves accuracy levels of 80-90% across a range of network conditions and operators.

Armed with the above, we present *our final contribution — the Peek-n-Sneak protocol — which allows cellular clients to "peek" into the channel cheaply and "sneak" in with their communication when the conditions are suitable*, i.e., when high throughput is predicted. Peek-n-Sneak is *incrementally deployable*, i.e., beneficial even to a single client that uses it. Furthermore, with wider adoption, *Peek-n-Sneak allows the set of participating clients in a cell to coordinate their communications, implicitly and in a fully distributed manner, with no support from the network operator*. By avoiding "stepping on each other's toes", each client is able to enjoy higher throughput, and complete its communication and go back to the idle state more quickly. In a sense, the coordination enabled by Peek-n-Sneak is akin to (though not the same as) that enabled by the widely-used carrier sense multiple access (CSMA) protocol in wireless LANs.

Our experimental evaluation of Peek-n-Sneak spans three network operators in two countries and includes both 3G and LTE networks. We find that a single stationary client employing Peek-n-Sneak for background downloads can save 20-60% of total energy consumed by a client that does not employ Peek-n-Sneak. The benefits grows as more clients adopt Peek-n-Sneak until the cell load is saturated. Further, we show that a mobile client employing Peek-n-Sneak can save on average up to 40% of total energy compared to not using Peek-n-Sneak. Finally, we show that a wider adoption of Peek-n-Sneak would also benefit foreground communication, by helping streamline and thereby reducing contention due to background communication.

2. CELLULAR LOAD DYNAMICS

We have performed extensive measurements spanning a period of four months, downloading 100+ GB of data over 100+ hours to characterize the dynamics of cellular load and its impact on mobile device energy consumption. We used TCP for all our experiments since operators have been known to treat UDP traffic differently (e.g., subject them to throttling, etc.). Our experiments were conducted on the following operator networks and locations:

- Airtel: 3G and LTE in Bangalore, India
- AT&T: LTE in Seattle, US
- BSNL: 3G in Bangalore, India

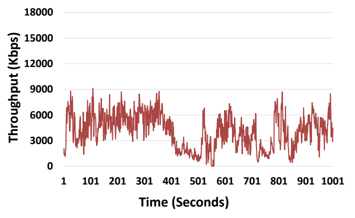
Since we do not have the ground truth on the cellular load, i.e., the total volume of data traffic due to the active users in the cell, we perform active throughput measurements to sense load. A higher throughput for our active download implies a lower cellular load due to other users, and vice versa. We postulate that any throughput variation is largely due to contention in the wireless channel, and not in the wired part of the operator network. In Section 3, we present evidence to corroborate this presumption. Note that in all cases, throughput was measured by downloading from a server located in the same geographic region as the cellular client, to avoid wide-area network bottlenecks.

To get a more detailed look at the dynamics of cellular load, we present the throughput measurements from a long, continuous TCP download of 1 GB on AT&T's LTE network in Seattle during peak hour (Figure 1a) and lean hour (Figure 1b), and 500 MB on BSNL's 3G network in Bangalore during peak hour (Figure 1c). The throughput samples were computed over 1-second intervals. The peak hour downloads were performed during a busy hour (midday) in downtown locations, when the cell can be expected to have many other active users. The figures show that throughput fluctuates considerably, with several periods of low throughput but also sustained periods (tens of seconds) of good throughput, despite it being a busy period. This suggests that there is considerable scope for benefiting from optimal scheduling over relatively short timescales.

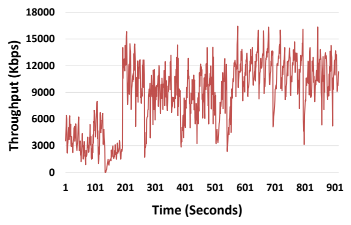
To analyze these results further, in Figure 2a, we show the autocorrelation of the throughput values for lags ranging from 0 to 50 seconds. The figure shows that, although the autocorrelation drops rapidly with increasing lag (which points to the fluctuating nature of cellular load and hence throughput), there is a modest degree of correlation over short time scales; for example, the autocorrelation exceeds 0.5 for about the first 10 seconds. This suggests that, even during peak hours, if we are somehow able to identify a good throughput period, then the chances are that we will continue to enjoy good throughput for next several seconds, enabling high-throughput and hence energy-efficient transfer of a few megabytes of data (e.g., a throughput of 1.5 Mbps over 10 seconds would amount to almost 2 MB of data).

Can signal quality serve as a predictor of good throughput periods? Figure 2b answers this question in the negative, showing that there is minimal correlation between link quality and throughput in a busy cell. This finding highlights the importance of devising a way to sense cellular load, so that there is hope of predicting good throughput periods.

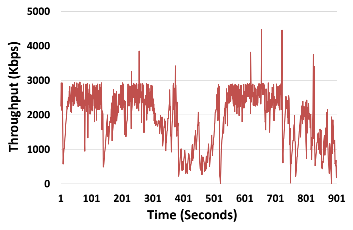
Finally, assuming we are able to accurately predict throughput, with low overhead, and thereby schedule background communication optimally, how much of battery energy savings can we expect? To answer this question, we compare the naive approach of downloading immediately, with an oracular approach of picking the optimal starting time for the download assuming perfect knowledge of future throughput. We used a workload comprising one 5MB download performed by a stationary device every 15 minutes. We treated 15 minutes as the "deadline" for completion of the download, thus giving the oracle some room to play with scheduling. Given the modest size of the individual downloads (5 MB), we assume that once the download starts, it runs to completion; this helps



(a) Throughput on AT&T LTE network during peak hour

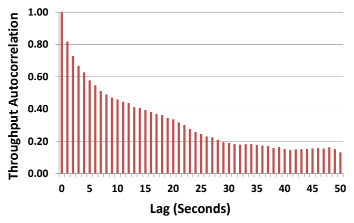


(b) Throughput on AT&T LTE network during lean hour

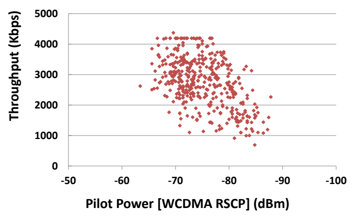


(c) Throughput on BSNL 3G network during peak hour

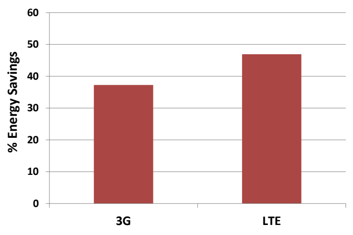
Figure 1: TCP download throughput over cellular networks



(a) Autocorrelation of throughput on AT&T LTE during peak hour



(b) Throughput vs Signal for BSNL 3G download



(c) Energy savings for an oracular approach

Figure 2: Analysis of cellular download throughput

avoid the overhead of radio "tails" [14]. Based on the throughput attained by these 5 MB downloads, we estimate the corresponding energy cost for the whole device (not just the radio).

Figure 2c shows the average energy savings of the oracular scheme relative to the naive scheme for BSNL's 3G network during peak hour in a busy downtown location in Bangalore as well as for peak hour traffic using AT&T's LTE network in Seattle. In both cases, we see significant energy savings of 37-47%.

The above findings clearly indicate that an effective throughput prediction scheme can be very beneficial. Note, however, that such a predictor is not sufficient, since multiple clients with background workloads could use the same predictor and start their downloads at the same time, to everyone's detriment. So, in our work, we develop low-overhead techniques for both cellular throughput prediction (LoadSense) and distributed coordination (Peek-n-Sneak).

3. LOADSENSE: SENSING CELLULAR LOAD

We now consider the problem of sensing cellular load in a way that imposes a low energy overhead on the mobile device and works without assistance from the network. The energy overhead requirement rules out performing active network measurements or even transitioning the radio to the high-power, active state. The lack of network assistance reflects the reality of cellular networks being closed, operator-controlled entities. Furthermore, unlike in the case of wireless LANs, the cellular interface on mobile devices does not support carrier sensing. Even snooping on background data traffic is not possible, as cellular modems do not decode data destined for other clients.

We look at how we can achieve carrier sensing like functionality for cellular interfaces, despite the challenges listed above. The basic intuition of our approach is that, as the number of active clients in a cell increases, the volume of downlink transmissions from the base station would likely also increase. As a result, the raw power in the downlink channel is likely to increase. This phenomenon would occur in both WCDMA (3G) and LTE networks, and points to how a client could sense the cellular load.

One issue is that we need to know the base power level, before we can determine whether there is an increase in power level due to load. To obtain the base power level, we can leverage pilot signals that are transmitted in cellular networks. Each cell tower periodically broadcasts pilot signals, to enable clients to detect the presence of the tower. Clients use pilot information for a number of purposes, such as cell reselection, handoff decisions, and synchronization. For instance, cell reselection is a procedure whereby clients can passively evaluate which tower to connect to in case there is a data transmission in the near future. This procedure is performed continuously, even when the radio is in the low-power, idle state.

The pilot power is a measure of how good the link is (i.e., the link quality) between the base station and the device. The link quality, as indicated by the pilot power, is an indicator of the download data rate that can be expected, in absence of background traffic. The closer the client is to the cell tower, the higher the pilot power it sees and the higher the modulation rate that can be used. On the other hand, the farther the device is from the tower, the lower the pilot power it measures and the lower the usable modulation rate.

However, the pilot power can sometimes give a conservative estimate of the achievable data rate, since it ignores transmit power control that is typically employed by cell towers while transmitting to individual clients. Depending on the power budget available at the base station, different amounts of power can be allocated for transmissions to different clients. If there are few other active clients in a cell, the base station can devote more of its power budget to serving a particular client. Thus, sometimes even when a

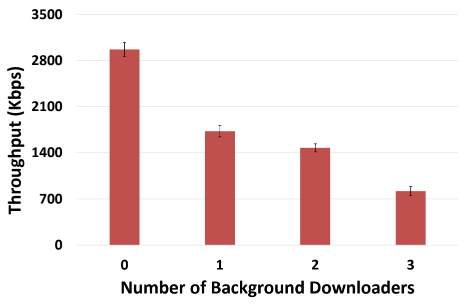


Figure 3: Throughput vs Number of Background downloaders (3G)

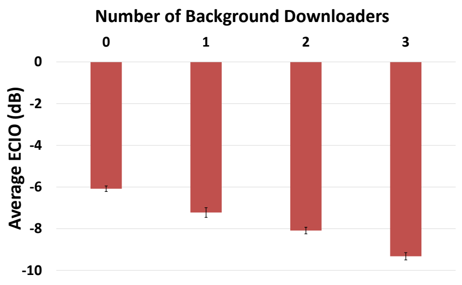


Figure 4: ECIO vs Number of Background downloaders (3G)

client is far away from the base station, it might still enjoy a high data rate if there are not many other active clients in the cell.

In addition to measuring the pilot power, mobile devices, in both WCDMA (3G) and LTE networks, measure the total, or raw, power in the channel. The raw power comprises the accumulation of the pilot power and the power due to data transmissions to other clients. The ratio of the pilot power to the raw (total) power in the channel turns out to be a a good indicator of cellular load. Thus, we define:

$PowerRatio = \frac{PilotPower}{TotalRawPower}$

In WCDMA (3G), the particular parameters that correspond to the pilot power and the power ratio are Received Signal Code Power (RSCP) and Received Energy per chip and Interference level (ECIO), respectively. In LTE, these are called Reference Signal Received Power (RSRP) and Reference Signal Received Quality (RSRQ), respectively. These parameters are typically exposed by most cellular devices through applications such as field test [2] [1]. We used a specific tool from Qualcomm, called QXDM, to obtain these parameters from a Windows Phone device. We can obtain around 3 samples a second from the cellular modem for each of these parameters.

When there are no transmissions, the power ratio is very close to 0 dB, since the raw power in the channel is equal to the pilot power observed, assuming that ambient noise is negligible.

With increased transmissions, the power ratio value drops, as the raw power in the channel is more than just that of the pilots in the carrier. There are couple of contributors to the increased raw power: (a) increased transmissions in the same cell, and (b) interference from neighboring cells, typically at locations near the cell edge. In either case, the expected throughput for the client device goes down. When there is a large number of active clients in the same cell, the proportional scheduler allocates fewer resource blocks to the individual clients due to increased contention, resulting in lower throughput per client. With increased interference from neighboring cells, the modulation rates have to be lowered to ensure successful communication, and thus the achievable bitrate for the client drops.

To validate point (a) above, we performed the following controlled experiment, which was done at a late night hour, to make it likely that the cell was unloaded. We had a test client measure the ECIO on a WCDMA channel for 20 seconds and then perform a 20-second download to measure throughput. Alongside this, we had 0 to 3 background clients, under our control and in the same cell, perform continuous downloads. Figure 3 shows that the throughput seen by the test client falls from 3 Mbps to under 1 Mbps, as the number of background downloads grows from 0 to 3. Correspondingly, the ECIO seen by the test client also drops (Figure 4) as the background load increases.

This controlled experiment shows that ECIO (or the power ratio, in general) is indicative of the load in the cell and hence the throughput that can be expected. However, both the link quality (as indicated by the pilot power) and the cellular load plus interference (as indicated by the power ratio) have an impact on the achievable data rate and hence the throughput for clients, as we discuss next.

To understand the relationship between the pilot power, power ratio, and observed throughput, we performed a large number of experiments, wherein we measured the RSCP and ECIO values on a 3G network while the device was in a disconnected state, and then initiated a data transfer immediately thereafter to measure throughput.

Figure 5 shows a 3D plot of throughput achieved at different locations and at different times of the day, corresponding to a range of measured values of RSCP and ECIO. We find that there is a wide range of throughput for each RSCP level, which indicates that congestion due to cellular load is a significant factor that impacts throughput.

For the purposes of throughput prediction, we confine ourselves to predicting one of two throughput classes — High or Low — where the threshold of 1.5 Mbps is used to separate the classes in 3G networks (7 Mbps in LTE networks). Figure 6 shows the RSCP and ECIO values for data points corresponding to the High and Low classes. It is clear that whenever RSCP is very low (<-90dBm) and/or ECIO is very low (<-10dB), we can predict with high accuracy (90+%) that a low throughput is to be expected. Similarly, when RSCP is very high and ECIO is low, we can predict with (90+%) accuracy that throughput is going to be high. However, we find that there is a significant intermediate region where the throughput could be high or low.

We observe similar behavior for LTE devices. Figure 7 show data points for the high and low throughput classes, and the corresponding pilot power (RSRP) and power ratio (RSRQ) values. The patterns are similar to those in 3G, with there again being an intermediate region where it is difficult to predict the high and low throughput classes.

It turns out that the power ratio (ECIO in WCDMA) exhibits high variation over time. The reason is the bursty nature of data traffic. At one instant, when lots of transmissions are occurring, ECIO would be low, and in the next instant, if there is a relative lull in transmissions, ECIO would go up again. Figure 8 shows a time series plot of ECIO measured at one client, where we had a separate client initiating short bursts of data transfer periodically, so create background traffic. We notice that ECIO variations are high whenever there is data transfer activity in the background.

This experiment clearly highlights the fact that there are temporal signatures in the ECIO (and also RSCP) parameters that could

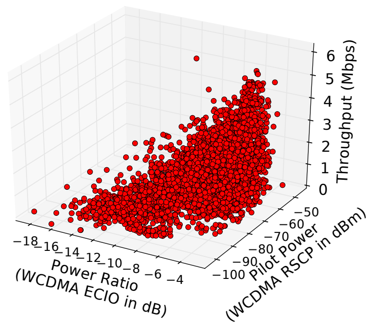


Figure 5: 3G Pilot Power vs Power Ratio vs Observed Throughput

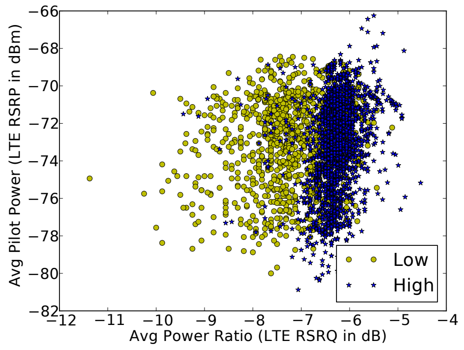


Figure 7: LTE Pilot Power vs Power Ratio for two throughput classes (many High points overlap with Low points)

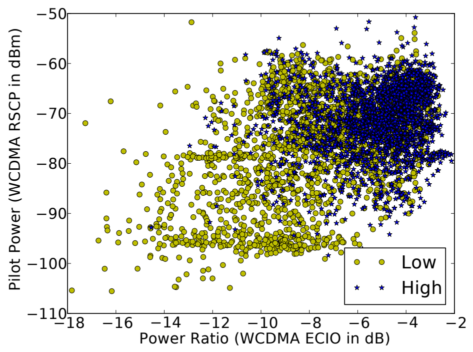


Figure 6: 3G Pilot Power vs Power Ratio for two throughput classes

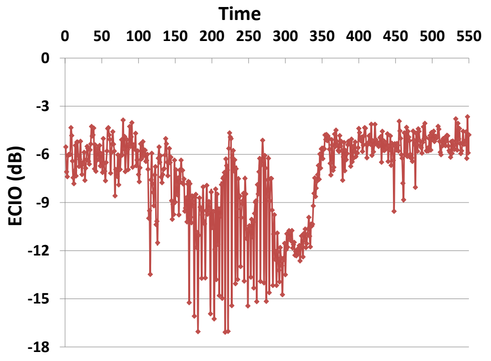


Figure 8: Temporal variation in ECIO with background traffic

be used to predict the throughput class better. To this end, we trained an empirical, Support Vector Machine (SVM) based classifier that is able to predict the occurrence of high or low throughput values, using the RSCP and ECIO measurements made by the client while it is idle, immediately preceding the download. We used the default Radial Basis Function (RBF) kernel for SVM. We trained using various features that capture temporal variations in ECIO and RSCP to determine which set of features enables the best prediction.

For evaluation, we tested each candidate empirical model with a dataset comprising equal numbers of high and low throughput observations, to avoid any bias. Accuracy for High (Low) period is computed as the fraction of actual High (Low) observations that we correctly predict as High (Low). Figure 9 shows the overall detection accuracy, and the accuracy for the individual classes, when various features are used as ingredients in the empirical model. By including as features the standard deviation, min, and max values of RSCP and ECIO over a short period (typically 3 seconds, although in Section 6 we evaluate other window sizes), the prediction accuracy is around 85% for predicting whether we will see good throughput. In other words, when we predict good throughput, we are right 85% of the time. However, we achieve a lower accuracy of 68% when predicting low throughput; in other words, when we predict low throughput, we in fact see high throughput 32% of the time. The reason for the lower accuracy is that our predictor is conservative, tuned to reduce misprediction of the high throughput class. This is important, since mispredictions of high throughput can cause the radio to be woken up and data transfer performed at an inappropriate time, leading to low throughput and wasted energy. On the other hand, mispredicting an actual high throughput period as low only leads to missed opportunities for downloading and hence increased delay, which is often less critical for background communication.

The predictions of high and low throughput based on the empirical model discussed above is what we term as *LoadSense*. We use LoadSense to make predictions of throughput class, and thereby infer the load conditions in a cell at runtime. Low throughput implies that the network is congested (Busy) and high throughput implies lightly loaded cell (Idle).

4. PEEK-N-SNEAK DESIGN

In this section, we describe the Peek-n-Sneak protocol that enables a large number of mobile devices that are connected to the same cell tower to coordinate their data transfers, without using any explicit message exchange.

Lack of support or visibility from the operator is a key challenge. One workaround might be to coordinate data transfers across different nodes using a central "controller" in the cloud. However, there are significant energy costs of doing so. Devices would need to continuously inform the controller of the current load conditions in their respective cells as well as keep the controller updated with their background transfer needs and deadlines. This will incur significant energy consumption on the client device for every such update, since the radio has to transition to high power active state (CELL_DCH state) before the device can communicate, even with the controller. Also, there would be a significant radio-tail energy cost, before the radio transitions back to the idle state.

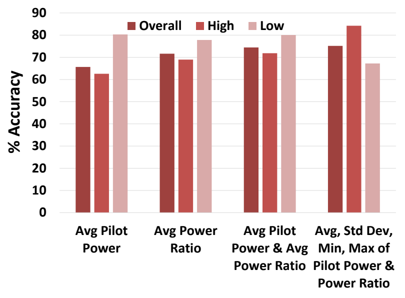
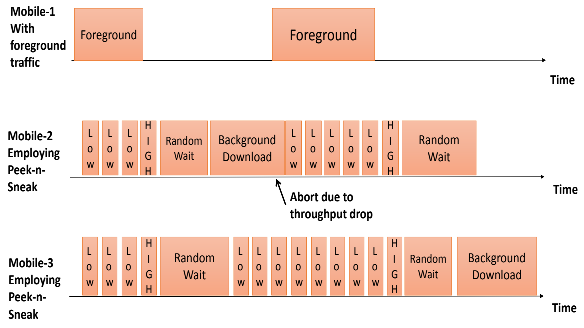


Figure 9: Predicting throughput using different features (3G)



Mobile-1 With foreground traffic
Foreground
Foreground
Time
Mobile-2 Employing Peek-n-Sneak
LOW
HIGH
Random Wait
Background Download
Abort due to throughput drop
Time
Mobile-3 Employing Peek-n-Sneak
Random Wait
Background Download
Time

Figure 10: Peek-n-Sneak Protocol Illustration

Instead, the Peek-n-Sneak protocol operates by implicit distributed coordination among the clients attached to a cell, thereby avoiding the energy costs of a centralized solution. The Peek-n-Sneak protocol operation is illustrated through an example in Figure 10. At a high level, the design is similar to the CSMA protocol used in WiFi. However, Peek-n-Sneak operates at much coarser timescales compared to packet-level coordination used in WiFi. For example, slot duration for sensing the channel using LoadSense is 3 seconds in Peek-n-Sneak compared to 9 μseconds used in WiFi. The second difference is that clients with user initiated foreground communication access the channel immediately and do not follow the Peek-n-Sneak protocol.

Whenever there is some data to be downloaded from the Internet but the download can be delayed (e.g., background updates to social networking applications), the request is sent to the background transfer service running locally on the mobile device. The client then performs LoadSense to determine whether the channel is idle or busy. When the channel is sensed as idle, the client starts a random backoff timer. Whenever a client is counting down idle slots, if the channel becomes busy due to either foreground communication or due to some other background client winning the random backoff contention, the counter is frozen as in WiFi. Finally, to ensure fairness and avoid starvation of other clients, after acquiring the channel for background transmission, each client voluntarily stops its transfer after it has transferred a maximum transfer unit (MTU) worth of data.

Collisions: Unlike WiFi, where only one client transmits at a time in a cell, cellular networks allows multiple clients to communicate simultaneously. For example, in LTE, time-frequency resource blocks are allocated to clients. When there are no other clients in the network, all resources can be allocated to one user. This is a case where link quality is what determines achievable throughput. However, with an increasing number of clients, the scheduler multiplexes available resources among the active clients. When two or more clients start downloading at the same time, each may experience a lower throughput than what it would have got if it were the only client operating in the cell. Thus, "collision" in Peek-n-Sneak merely implies sharing of the channel that results in lower throughput, rather than the total loss of throughput as is typical in WiFi collisions.

However, we still would like to avoid the lower throughput due to sharing. So, during download, Peek-n-Sneak at a client continues to monitor the throughput that the client receives. If the client detects that its throughput is lower than predicted (say, because of another client initiating a foreground transfer or collisions among background clients), it has the option to suspend its background transfer. Before doing so, the client estimates the energy consumption for the ongoing data transfer, if the download were continued to completion at the current throughput, and compares it with the energy cost of suspending the transfer (thereby incurring the radio-tail energy cost) and resuming it at a later point in time when the expected throughput is higher. If the energy savings is higher than a certain threshold for the suspend-and-resume option, the client pauses the download and goes back into sensing mode to determine when it should resume.

Finally, Peek-n-Sneak not only coordinates background clients, it also ensures that background traffic does not overlap with foreground traffic. Thus, foreground communication also enjoys better performance since they have to contend with fewer clients for resources. One exception to this conservative policy is that when there is already a background transfer that is active at the point when one or more foreground clients start downloading, the background transfer continues unless it experiences a significant drop in throughput because of (significant) contention with the foreground communication.

5. IMPLEMENTATION DETAILS

We describe two engineering challenges we faced while implementing our system. First, from an energy efficiency viewpoint, LoadSense should ideally be implemented inside the cellular modem as it would avoid the energy cost of waking up the CPU periodically to poll the radio for obtaining channel measurements. The cellular modem can perform an asynchronous callback to the CPU whenever the channel is idle. However, since current cellular modems do not support such a callback feature, our implementation uses CPU-based polling to query the modem. Second, the frequency of polling the channel parameters impacts how quickly and accurately LoadSense can estimate channel load. We explored two ways to poll the modem, each of which entails a different polling frequency.

Our first version of the system was completely built on the mobile client. We used the Radio Interface Layer (RIL) API calls to query the modem to obtain pilot power and power ratio values. However, the current RIL API does not provide fine-grained samples, and we observed that, irrespective of how frequently we invoke the API, the values only changed every two to three seconds. We evaluated LoadSense using this setup (see section 6), and observed that the accuracy is quite low (<70%). To address this issue we employed a different setup to measure pilot power and power ratio using a desktop tool provided by the modem vendor. For both 3G and LTE equipped client devices, we used a Windows Desktop tool called QXDM from Qualcomm. Our 3G experiment setup consisted of multiple phones running the Windows Phone 7.5 OS, each tethered to a separate Windows laptop running the QXDM tool. For LTE, since there are no LTE capable phones currently supported in the LTE network in India because of a mismatch in spectrum band, we used multiple LTE USB dongles for the experiments. We can read the power ratio and pilot power values for both 3G and LTE interfaces in realtime at a high frequency (5 per second in the radio idle state).

We also implemented a background transfer service (BTS) app that takes requests for downloading data with an optional deadline parameter. When a new background request is in the queue, the BTS app employs the Peek-n-Sneak protocol to determine when to initiate the data transfer. To perform LoadSense, the BTS app requests the desktop application for pilot power and power ratio values for a certain period of time and then predicts whether the channel is busy or idle based on the expected throughput class.

6. EVALUATION

In this section, we evaluate the LoadSense primitive and the Peek-n-Sneak coordination protocol. All experiments are conducted over commercial cellular networks in Bangalore (BSNL 3G Network and Airtel LTE network). We use the HTC Maaza phone running the Windows Phone 7.5 OS for the 3G experiments and the Huawei E392 USB dongle for the LTE experiments.

We first start with micro-benchmarks to evaluate the accuracy of our LoadSense technique. We then demonstrate the effectiveness of Peek-n-Sneak through several experiments in both static and mobile settings, over both 3G and LTE networks.

6.1 Micro-benchmark

In order to evaluate the accuracy of LoadSense, we need to compare the LoadSense prediction (high/low) with the observed cellular throughput at a given mobile. However, since LoadSense works when the radio is in RRC disconnected state, we don't know the exact cellular throughput available when the radio is passively estimating load. Therefore, as soon as the LoadSense technique comes up with a prediction, we perform an active TCP download (for 20 seconds) immediately after the prediction and measure the actual throughput achieved and use this as our ground truth. While the ground truth could change between the prediction and measurement intervals, this is mitigated in our experiments since 1) as we discuss below, our prediction interval is fairly short (3 seconds) and 2) the autocorrelation findings in Section 2 indicate that the available throughput typically remain stable over ten seconds or more.

We classify measured cellular throughput below a certain threshold (unless specified otherwise, we used throughput of 1.5 Mbps in our 3G experiments and 7 Mbps in our LTE experiments) as low, and as high otherwise. Similarly, LoadSense also makes a low or high prediction based on its observations. As mentioned earlier, the detection accuracy of LoadSense is computed as the fraction of measured throughput values classified as High (Low) that are predicted accurately by LoadSense as being High (Low) and use this as our metric for evaluating the effectiveness of LoadSense.

Finally, note that signal quality (pilot power) is also automatically taken into account in our load estimation. LoadSense predicts the throughput available as being low when the pilot power at the mobile radio is poor and/or when there is high amount of congestion due to competing mobile traffic.

6.1.1 *Detection time*

We first evaluate how long a client needs to observe the channel before being able to make an accurate prediction. The observation window plays a role in how quickly we can predict load changes. Moreover, detection time also determines the length of the slot used in Peek-n-Sneak protocol.

Figure 11 shows overall accuracy of predictions for both high and low throughput classes for different observation windows. When the observation window is less than 3 seconds the number of samples available for power ratio parameter and pilot power are few, leading to low accuracy (65%). The accuracy steadily improves to 75% with an increase in the size of the observation window mainly owing to increased number of observations available for LoadSense. However, accuracy does not improve significantly beyond 3 seconds, which suggests that a 3-second observation of channel condition is sufficient to capture most of the load characteristics in a cell and predict the throughput class for the next few seconds effectively. Also, it is preferable to have a short detection time in order to keep the random backoff overhead of Peek-n-Sneak low. We observed similar behavior on LTE networks. For the rest of our experiments we choose a detection time of 3 seconds for both 3G and LTE.

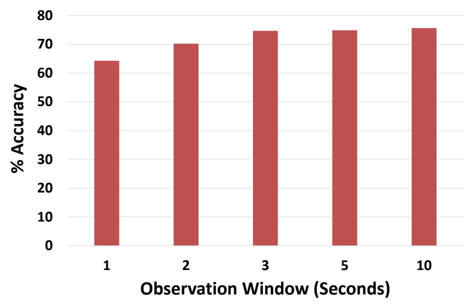


Figure 11: Detection time vs Accuracy for 3G

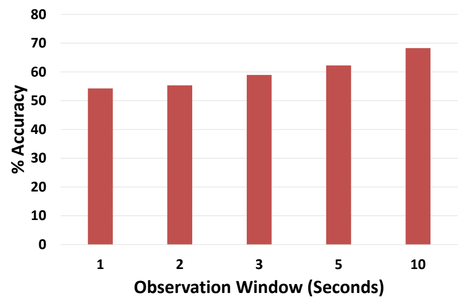


Figure 12: Detection time vs Accuracy for 3G using RIL APIs

We also evaluated the detection time needed when using the RIL APIs to poll the modem directly from the phone. Figure 12 shows the prediction accuracy for different observation windows. The main issue with the current implementation of RIL is that it provides very coarse grained samples. We find that even at 10 second interval, the accuracy achieved is still below 70% due to very few samples of channel measurements. In addition, a large detection time elongates the slots used in the Peek-n-Sneak protocol and reduces efficiency. Thus, we employ the desktop-assisted setup to obtain fine-grained channel measurements for the rest of the experiments. Note that this issue is not an inherent bottleneck since the modem is able to provide fine-grained measurements through the tool. This just shows that the current set of RIL APIs is not well suited for our needs.

6.1.2 *Throughput Threshold*

Next, we evaluate the sensitivity of detection accuracy with respect to the throughput threshold parameter that is chosen for classifying the low and high throughput classes. Figure 13 shows the accuracy for different threshold values. For 3G networks, we ob-

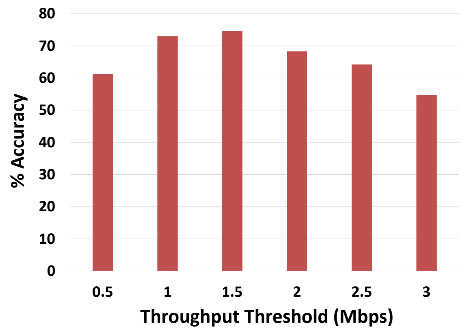


Figure 13: Throughput threshold vs Accuracy for 3G

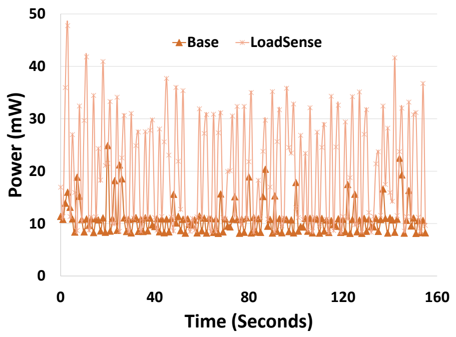


Figure 14: Power draw with and without LoadSense

served that accuracy peaks at around 1.5 Mbps throughput. For values both above and below this threshold the accuracy is lower. A deeper look at the dataset revealed that 1.5 Mbps is close to the median value for the throughputs we observed in the 3G network. The SVM based technique in LoadSense is able to learn a good classifier when the threshold value splits the dataset into equal sized classes. In LTE experiments we observed the throughput threshold value which achieves best accuracy is 7 Mbps. While we can improve the accuracy for different throughput thresholds by tuning the classifier or using a different classifier, we believe using the median throughput to separate the high/low classes is a reasonable operating point for a primitive such as LoadSense.

For the rest of the experiments, we choose 1.5 Mbps for 3G and 7 Mbps for LTE as the thresholds respectively for classifying busy and idle periods. We consider the channel to be Busy (Idle) when predicted throughput is Low (High).

6.1.3 Energy Consumption

Energy consumption has two components – (1) Sensing energy spent running LoadSense continuously in the background and (2) Transfer energy spent for the actual data transfer as well as overheads (wakeup and tail). Due to the engineering constraints of our system implementation, we used a combination of two setups to estimate total energy. To compute sensing energy, we used the mobile only implementation of our system which invokes RIL call periodically (once per second) to capture the computational overhead. Since the desktop assisted implementation achieves the best accuracy, we use this to monitor the radio on time and RRC state changes and use these values to estimate the transfer energy spent (including the CPU energy of the background transfer with the display switched off) during the course of an experiment. We then compute the sum of these two components to estimate the total energy spent. Note that, if the cellular modem implemented the LoadSense primitive, the cost of the first component can be eliminated.

Figures 14 shows a snapshot of power drawn by the device over time without LoadSense and with LoadSense, respectively. The average power draw increases only by 12mW from 10mW to 22mW corresponding to sensing overhead of running LoadSense continuously. In the experiments to follow, for computing total energy cost of Peek-n-Sneak we add both the transfer energy as well as the above sensing energy cost (12mJ).

6.1.4 Location Sensitivity

In addition to just measuring load in a cell with background traffic that already exists, we also looked at settings when we used our own clients to generate traffic in a controlled setting. The objective is to determine how well LoadSense can be used to detect the presence or absence of synthetic background traffic at different locations inside a cell. In addition, this analysis forms the basis for the Peek-n-Sneak protocol which uses LoadSense to implicitly inform the outcome of a contention round to other clients running LoadSense. We used one mobile client referred to as interferer to periodically download data for 60 sec and remain idle for 60 sec. Another mobile client referred to as observer runs LoadSense continuously to detect whether the channel is busy or idle. Note that in this experiment, we know for sure that the channel is loaded when interferer is active. However, since we do not have control over other users in a cell, there can be times when interferer is idle, but the channel could still be busy.

We vary the locations of the observer while keeping the interferer in one location inside an office building. The distance between the clients are as far apart as 800 meters at some locations, where the observer is outdoors. During each experiment we ensured that both clients were attached to the same cell tower. At each location we run LoadSense every 3 seconds for a duration of 10 mins, which accounts for a total of 200 predictions per location. Figure 15 and 16 depict the accuracy of predictions using LoadSense at four different locations using 3G and LTE, respectively. At each location, accuracy is shown for two periods – times when the interferer is on and times when the interferer is off. Overall, LoadSense detects presence of background interferer accurately more than 80% of the time irrespective of the location of the observer in both 3G and LTE networks. The accuracy drops by 10-20% in the case of idle period prediction but this could be due to lack of accurate ground truth information for classifying idle periods (there could be other active users attached to the cell that we are unaware of). This experiment shows that the detection accuracy of LoadSense is high enough to be useful for coordinating background data transfers. Each client running Peek-n-Sneak can use LoadSense to coordinate their contention, since they can effectively detect each others downloads.

6.2 Macro-benchmarks

We evaluate Peek-n-Sneak using multiple phones using a variety of synthetic and real workloads.

To evaluate Peek-n-Sneak in a typical multi-phone setting, we setup 6 stationary phones connected to the same cell and performed a combination of experiments with foreground and background traffic. The objectives of the experiment are twofold – (1) to evaluate the impact of the Peek-n-Sneak protocol on energy consumption and latency, and (2) to see how well Peek-n-Sneak performs in a network where other clients do not employ Peek-n-Sneak. A client may not employ Peek-n-Sneak if it is either performing foreground

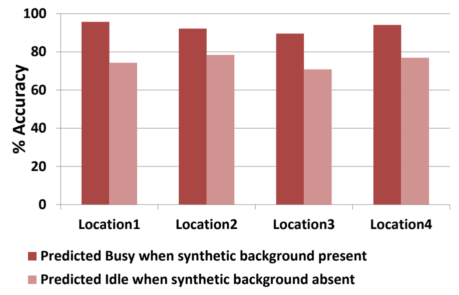


Figure 15: Detecting synthetic background traffic from different locations in a 3G cell

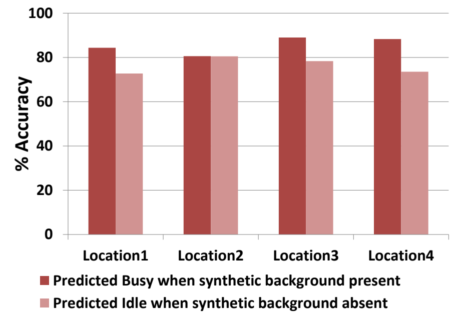


Figure 16: Detecting synthetic background traffic from different locations in a LTE cell

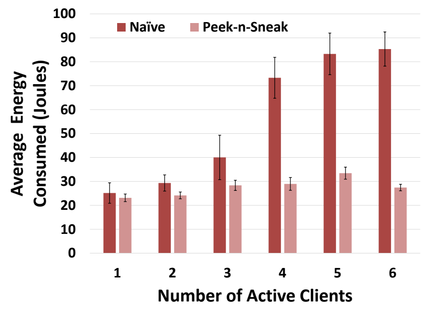


Figure 17: Average total energy consumption for Naive and Peek-n-Sneak with increasing load

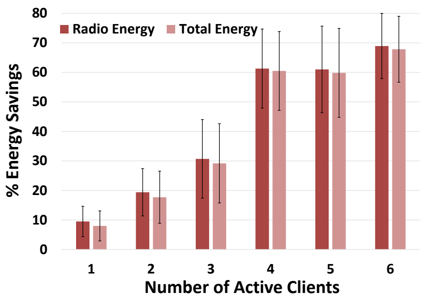


Figure 18: % savings for Peek-n-Sneak with increasing load consisting of only background downloads

transfers which cannot be time-shifted or if it does not implement Peek-n-Sneak. The latter setting captures the benefits of Peek-n-Sneak during an incremental deployment of the protocol.

6.2.1 *Multiple Phone Experiments*

For these experiments, we emulate a periodic syncing background workload where each phone downloads 5 MB of data from an Internet server every 15 mins. We ensure that downloads across the clients are not synchronized since in realistic settings the syncing schedules of individual phones are independent of each other. In the beginning of the experiment, every client sleeps for a random duration of time (up to 4 mins), before initiating a sync request for download. In the first set of experiments we look at a *homogeneous* setup where all clients are using Naive or all are using the Peek-n-Sneak protocol. Naive clients basically download data as soon as the request is generated (as is the common practice today), while in Peek-n-Sneak, the clients download data according to the protocol with a deadline set to 15 mins (if a low load opportunity does not arise, the download is triggered anyway to ensure completion within 15 mins). The numbers in the plots are an average of 5 runs. Due to complexity of settings up multiple clients, experiments were conducted over several days at different times of the day.

We vary the number of active clients in each experiment from 1 to 6 and run either Naive or Peek-n-Sneak on each client. In each configuration, we performed 5 runs and computed the average metrics over these runs. Figure 17 shows the average total energy in Joules consumed by Naive and Peek-n-Sneak for different number of active clients. As mentioned earlier, we include the compute overhead of running LoadSense while the client is waiting for the right opportunity to download. First, we see that as the number of active clients increase, energy consumption for each client increases significantly for the Naive approach. This is due to increased contention between different downloading clients. Peek-n-Sneak implicitly serializes downloads and each client gets to access the channel without contention from other active clients. Thus, energy consumption only increases by 23% when going from 1 to 6 active clients using Peek-n-Sneak, while with the Naive approach, the corresponding energy increase is as high as 340%. Energy savings come at the cost of increased latency. The average latency (Figure 19) to download is higher with Peek-n-Sneak, but remains within 3 mins of when the sync request is generated. Moreover, for background workloads, which are not initiated by the user, a small increase in latency may not impact user experience.

Figure 18 shows the percentage savings for Peek-n-Sneak over Naive for radio energy and total energy. The compute overhead of running LoadSense and Peek-n-Sneak is very low and does not impact total energy savings. Average energy savings go up to 62% in the 6 client experiment, mainly owing to reduced contention and also since LoadSense ensures that the expected throughput is high, before initiating a download, thereby resulting in shorter download time. In the experiments, we find that download throughput in Peek-n-Sneak is 150% higher compared to Naive with 6 clients. However, the improved throughput gains do not translate to equivalent energy savings mainly due to the cost of aborted transfers (and the associated overheads such as radio tails). Since there can be background data transfers that are not under our control, Peek-n-Sneak clients stop the data transfer as soon as the throughput drops below the low threshold and then again looks for periods when it can download. Every interruption causes additional energy due to radio wakeup and tail overheads.

Next, we perform experiments with a combination of Naive and Peek-n-Sneak clients. The Naive clients can be seen either as foreground clients that want the data downloaded right away or as background clients that do not implement Peek-n-Sneak (corresponding to an incremental deployment scenario).

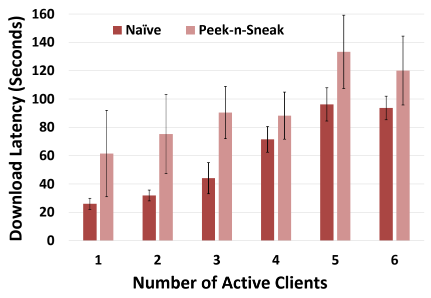


Figure 19: Average download latency for Naive and Peek-n-Sneak with increasing load consisting of only background downloads

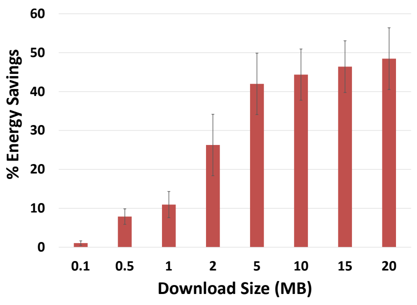


Figure 21: % savings for Peek-n-Sneak for different download sizes

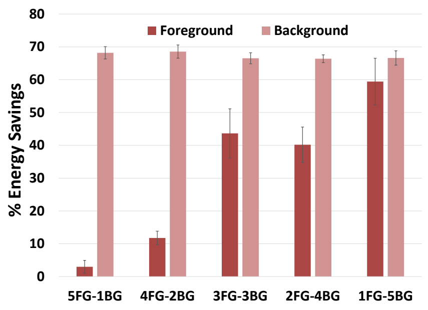


Figure 20: % savings for Foreground and Peek-n-Sneak clients for different Foreground-Background client combinations

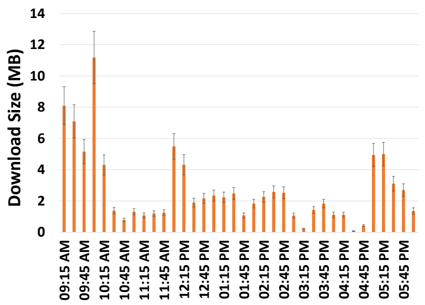


Figure 22: Average download size in MB per 15 min sync interval during work hours for Email and RSS feed reader applications

Similar to the previous setup we have 6 clients, and we choose different combinations of clients doing foreground traffic (Naive) and clients doing background traffic (Peek-n-Sneak). Figure 20 shows energy savings for foreground clients and background clients separately compared to the case where all 6 clients follow Naive irrespective of whether they are doing foreground or background traffic. The first thing to note here is that even in a scenario where there is only 1 client following Peek-n-Sneak and 5 naive clients performing foreground traffic, we achieve savings as high as 68% for the Peek-n-Sneak background client. The energy savings remain stable above 65% even as the number of background clients increase. This is due to implicit coordination achieved by Peek-n-Sneak which serializes the data transfers. A beneficial side-effect of Peek-n-Sneak is that even clients with foreground traffic achieve energy savings as a result of reduced contention from the Peek-n-Sneak clients. As the proportion of background clients running Peek-n-Sneak increases, energy savings for foreground clients steadily improves, reaching as high as 60%. This experiment shows that Peek-n-Sneak clients can coexist and achieve savings even in networks where other clients may not be following the protocol, as is typically the case during incremental deployment of new protocols.

6.2.2 *Workload Size*

So far we only looked at fixed size background workloads. We now evaluate the impact of varying background workload size on the savings achieved by Peek-n-Sneak. In this scenario, we had 1 phone downloading the specified workload using Peek-n-Sneak while we had 5 other phones performing foreground traffic in the same cell (similar to the heterogeneous setup in the previous experiment with 6 clients).

In this experiment, we vary the download size over a wide range of values that correspond to different real-world background workloads (email sync, app updates and music downloads). We repeat each experiment 10 times using both Naive and Peek-n-Sneak approaches to perform the download using the 3G network and measure the average savings of Peak-n-Sneak over Naive. Figure 21 shows download size varying from 100 KB to 20 MB and the corresponding average total energy savings achieved by Peek-n-Sneak. For small download sizes (<1 MB) we find that radio overheads (wakeup and tail) dominate the energy cost of the actual data transfers, resulting in negligible energy savings. We find that savings steadily increase with download sizes up to a certain point (5 MB) and then saturates (46%). The reason for the saturation is that large background downloads are often interrupted by presence of foreground traffic. While the Peek-n-Sneak protocol avoids downloading during these periods, it still ends up paying the cost of putting the radio to sleep (tail overhead) and waking it up, thereby limiting the energy savings.

6.2.3 *Real Workloads*

In this experiment, we collected real world traces for two background applications – email and RSS feed reader – that are configured to sync every 15 mins. The RSS reader is setup to download feeds from popular news sites and blogs. Figure 22 shows the average download sizes during each 15 min interval averaged over two separate days from 9am to 6pm. We replayed the traces with Peek-n-Sneak and computed energy savings. Again, as before, we had five other phones generating foreground traffic.

Figure 23 shows the savings achieved during each sync interval for Peak-n-Sneak as compared to Naive. We find that Peek-n-Sneak achieves an overall energy savings of 29% for the background workload averaged over two days.

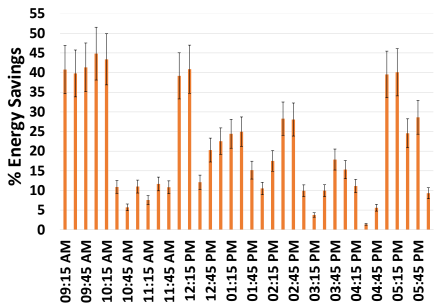


Figure 23: % savings for Peek-n-Sneak per sync in 3G

6.2.4 *Mobile Experiments*

In these experiments, we apply LoadSense in mobile settings as well where throughput variation is due to a combination of both signal and load effects. Since the mobile client remains in a cell for a short period of time and channel conditions differ from cell to cell, we disabled the random backoff in the Peek-n-Sneak protocol. We simply employ LoadSense to identify periods when the throughput is expected to be high to initiate a data transfer. We abort the transfer when throughput falls below the default threshold. We compare this scheme with the Naive scheme that simply downloads when the request is generated. In addition, we also compare this with a scheme that uses only signal to predict high throughput periods. Prior work [15] has shown that signal correlates well with expected throughput. We performed 25 mobile experiments while driving around in different parts of Bangalore at various times of the day on BSNL 3G network. An experiment consisted of running Naive, Signal-only and Peek-n-Sneak schemes separately on a particular route while downloading 10MB of data in 30 mins(two back-to-back 5 MB sync workloads).

Energy savings for Signal-only scheme and Peek-n-Sneak over Naive averaged over 25 runs is shown in Figure 24. Peek-n-Sneak achieves an average total savings of 45% across these runs while Signal-only achieved 23% savings. Thus, load awareness leads to a doubling of the energy savings compared to the Signal-only scheme. This is because often there are times when the signal is quite good, but the cell is congested. Peek-n-Sneak is able to avoid downloading data until both load and signal conditions are good, thereby increasing energy savings.

Figure 25 shows the average time (in seconds) when the download finished for the three schemes. Clearly latency increases for both schemes compared to Naive as a result of waiting for the right time to download. Latency for Peek-n-Sneak is the highest (2.3x of Naive) as it is more conservative in downloading data. However, in absolute terms, the additional delay is only about a couple of minutes, which may be acceptable for background data transfers.

7. DISCUSSION

Other applications. Besides optimizing energy for background applications, LoadSense can be applied in other settings. For example, the typical signal bars shown on a phone indicate merely the signal strength but does not take into account interference or congestion in the cell; LoadSense can be used to provide an indication of the expected throughput for a foreground application, which might be a more meaningful indicator for the user. Also, LoadSense can be used to estimate the number of people in a region.

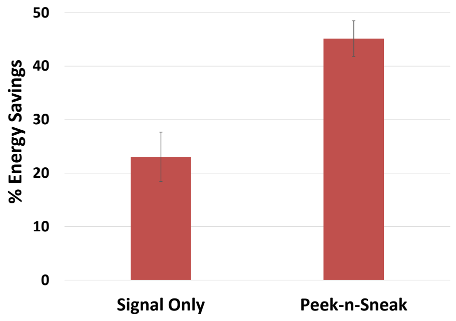


Figure 24: % energy savings for Peek-n-Sneak and Signal-only in mobile settings

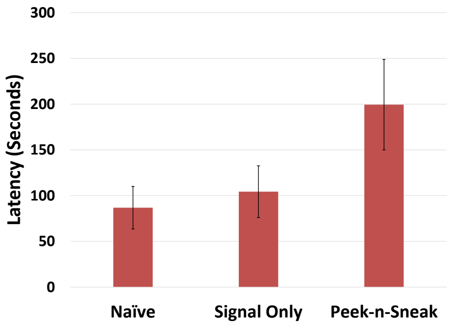


Figure 25: Average download latency for Naive, Signal-only and Peek-n-Sneak

For example, one can estimate road traffic conditions by correlating it with number of active users in a cell.

Multi-user MIMO. LTE standard allows multi-user MIMO where multiple antennas at the base-station can transmit different streams to different clients at the same time. A mobile client may observe total power in the channel increasing while available throughput does not decrease until all streams at the base-station is exhausted. LoadSense for such networks will require dynamic calibration to learn the relation between interference levels and expected throughput in such cells. However, in our experiments, we did not observe such behavior. Perhaps this is because most LTE deployments today use Frequency Division Duplex (FDD) which separates uplink and downlink transmissions on different frequency bands and implementing MU-MIMO on FDD systems is challenging.

Changing cellular standard to broadcast load information. Clearly, the base station has the best knowledge of load levels in a cell and if cellular standards allow the base station to broadcast this information, LoadSense at the mobile clients would not be necessary (though Peek-n-Sneak would still be necessary for coordination among different nodes). However, even if the standards supported such a feature, LoadSense may still be needed as operators may not be willing to continuously broadcast cell load information for competitive reasons.

8. RELATED WORK

Cellular measurement studies. Several papers have examined the performance and other characteristics of cellular networks using measurements [12, 13, 17]. Study of spatio-temporal characteristics of base station load in [13] show low correlation between neighboring base stations and significant temporal changes in base station load. These characteristics corroborate our measurement study and imply that time- and space-shifting techniques, as employed in this paper for the static and mobile scenarios, will likely be effective in "water-filling" unused cellular capacity using background traffic.

Throughput estimation Throughput estimation is typically performed using active measurements, albeit using as few packets as possible. For example. for estimating throughput on wired networks, there is a long list of research papers and tools available [3, 4, 7, 8, 16]. Other work have extended these techniques and developed new tools to estimate throughput over 802.11 standard-based wireless LANs [10, 11].

However, to our knowledge, there does not exist any tool to date that can accurately estimate throughput in cellular networks. In fact, authors in [9] show that conventional bandwidth estimation tools are ineffective in estimating throughput in cellular networks. Furthermore, in order to save energy, we require *passive* estimation of throughput when the cellular radio is in the RRC disconnected state. Even sending a few packets to estimate cellular load can result in significant drain in energy, thereby wiping out the gains of such a load sensing approach. Therefore, instead of estimating throughput, we use signal and interference power measurements of the cellular radio to simply determine if the cell is loaded or not. This is analogous to carrier sensing used in WiFi and exploited by systems such as Idle Sense [6] to optimize throughput.

Cellular energy optimization Bartendr [15] is a system that uses predicted mobility and future signal strength values to preferentially schedule communication when the signal strength is high, thereby reducing energy consumption due to mobile communication. However, Bartendr does not account for the current load in a given cell. Therefore, preferentially scheduling communication at a high signal strength alone may not be enough for reducing energy consumption if the current cell is highly loaded. In this paper, we show that incorporating load sensing in mobile settings (which also encompasses signal-based scheduling since load estimation takes mobile's signal strength into account) can result in significant energy savings. There are also techniques that optimize radio tail energy [14] by ensuring that transmissions from applications are coordinated but these energy optimization are complementary and additive to the energy savings obtained by scheduling transmissions when the cellular network is lightly loaded.

9. CONCLUSION

Obtaining good throughput is key to reducing the energy consumption of background communication over cellular networks. We have shown experimentally that throughput depends not only on the link quality but also on the cellular load. Given this, we have developed the *LoadSense* technique, to obtain a measure of cellular load and predict the expected throughput class, while incurring little overhead. In turn, LoadSense enables the *Peek-n-Sneak* protocol, which allows a set of devices to implicitly coordinate their (background) communications, with a view to benefitting them all in terms of higher throughput and lower energy drain.

On a broader note, cellular networks are reputed to be closed, operator-controlled entities. Even so, our work shows that it is possible for end devices to glean information indirectly and be smart about how they communicate, for the benefit of both the devices and the network. Such a win-win proposition could nudge operators towards greater openness.

10. ACKNOWLEDGEMENTS

We thank our shepherd, Thyaga Nandagopal, and the anonymous reviewers for their constructive comments. We thank Ajay Manchepalli for helping us collect measurements in US.

11. REFERENCES

[1] iPhone 4s Review. http://bit.ly/uLXlUm.

[2] Samsung Galaxy S2 Review. http://bit.ly/qjMdYv.

[3] C. Dovrolis, P. Ramanathan, and D. Moore. Packet-dispersion techniques and a capacity-estimation methodology. *IEEE/ACM ToN*, 12(6):963–977, 2004.

[4] A. Downey. Using pathchar to estimate internet link characteristics. In *In Proceedings of ACM SIGCOMM*, 1999.

[5] H. Falaki, D. Lymberopoulos, R. Mahajan, S. Kandula, and D. Estrin. A first look at traffic on smartphones. In *Proceedings of the 10th annual conference on Internet measurement*, pages 281–287. ACM, 2010.

[6] M. Heusse, F. Rousseau, R. Guillier, and A. Duda. Idle sense: An optimal access method for high throughput and fairness in rate diverse wireless lans. In *ACM SIGCOMM*, 2005.

[7] N. Hu and P. Steenkiste. Evaluation and characterization of available bandwidth probing techniques. 21(6), 2003.

[8] M. Jain and C. Dovrolis. End-to-end available bandwidth: measurement methodology, dynamics, and relation with tcp throughput. In *In Proceedings of ACM SIGCOMM*, 2002.

[9] D. Koutsonikolas and Y. C. Hu. On the feasibility of bandwidth estimation in 1x evdo networks. In *In Proceedings of the ACM Mobicom International Workshop on Mobile Internet Through Cellular Networks (MICNET 2009)*, September 2009.

[10] K. Lakshminarayanan, V. N. Padmanabhan, and J. Padhye. Bandwidth estimation in broadband access networks. In *ACM/USENIX Internet Measurement Conference*, October 2004.

[11] M. Li, M. Claypool, and R. Kinicki. Wbest: a bandwidth estimation tool for ieee 802.11 wireless networks. In *In Proceedings of IEEE LCN*, 2008.

[12] X. Liu, A. Sridharan, S. Machiraju, M. Seshadri, and H. Zang. Experiences in a 3g network: interplay between the wireless channel and applications. In *MobiCom*, 2008.

[13] U. Paul, A. Subramanian, M. Buddhikot, and S. R. Das. Understanding traffic dynamic in cellular data networks. In *In Proceedings of IEEE INFOCOM*, 2011.

[14] F. Qian, Z. Wang, A. Gerber, Z. Mao, S. Sen, and O. Spatscheck. TOP: Tail Optimization Protocol for Cellular Radio Resource Alocation. In *Proceedings of the eighteenth IEEE International Conference on Network Protocols*, Kyoti, Japan, October 2010.

[15] A. Schulman, V. Navda, R. Ramjee, N. Spring, P. Deshpande, C. Grunewald, K. Jain, and V. N. Padmanabhan. Bartendr: A practical approach to energy-aware cellular data scheduling. In *Proceedings of the 16th Annual International Conference on Mobile Computing and Networking (Mobicom)*, Chicago, Illinois, September 2010.

[16] J. Strauss, D. Katabi, and F. Kaashoek. A measurement study of available bandwidth estimation tools. In *In Proceedings of ACM Internet Measurement Conference (IMC)*, 2003.

[17] W. L. Tan, F. Lam, and W. C. Lau. An empirical study on 3G network capacity and performance. In *INFOCOM*, May 2007.