EnVi: Energy Efficient Video Player for Mobiles

Sahil Suneja
University of Toronto, Canada
sahil@cs.toronto.edu

Vishnu Navda
Microsoft Research India
navda@microsoft.com

Ramachandran Ramjee
Microsoft Research India
ramjee@microsoft.com

Eyal de Lara
University of Toronto, Canada
delara@cs.toronto.edu

ABSTRACT

Watching online movies and TV shows while on the go is rapidly becoming one of the most common uses of smartphone. Unfortunately, sustained use of the cellular interface for long-standing data transfers severely reduces the battery life of the device. This paper introduces EnVi, a signal-aware scheduler that saves energy for mobile downloads by scheduling data transfers when signal quality is expected to be good, and avoiding communication in other periods. EnVi uses a self-calibrating approach that does not require assistance from the network, or knowledge of the user's route and future signal profile. EnVi is robust to signal profile changes due to time of day effects, different cell tower associations and new cell tower deployments. In a preliminary evaluation on a suburban train route, our EnVi video player achieved average radio energy savings of 20.7%, going upto as high as 32.5%.

Categories and Subject Descriptors

C.2.1 [**Computer-Communication Networks**]: Network Architecture and Design—*Wireless communication*; C.5.3 [**Computer System Implementation**]: Microcomputers—*Portable devices*; B.8.0 [**Performance and Reliability**]: General

General Terms

Algorithms, Design, Experimentation, Measurement, Performance

Keywords

Energy; Mobile; Video; 3G; HSPA; Fast Dormancy

1. INTRODUCTION

With near-ubiquitous availability of high speed cellular internet access, mobile users are becoming extensively involved in streaming online music and radio (e.g. via Pandora), and watching online movies, and TV shows (e.g. Youtube, Netflix) on their smartphones and tablets while on the go. However, sustained use of the cellular interface for long-standing data transfers can reduce the battery life of the device significantly, for example a typical smartphone with a fully charged battery only lasts around 3 hours while viewing a YouTube video stream [5].

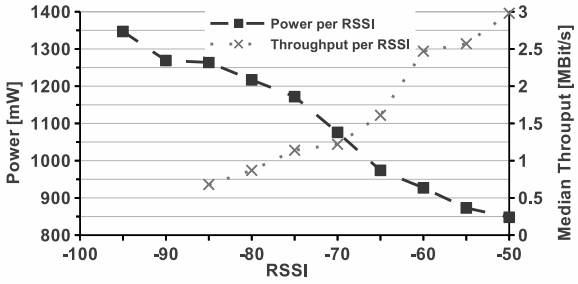

Figure 1: Power and Throughput vs. RSSI

Furthermore, under mobility, when radio communication is carried out during periods of poor signal quality, energy per bit transferred is almost a factor of 6x higher compared to that at good signal conditions owing to a combination of lower data rates and increased power draw (figure 1). In a mobile setting, by sensing the operational environment, there is an opportunity to reduce energy drain by scheduling data transfers when signal quality is expected to be good, and avoiding communication in other periods. This technique is particularly suitable for applications that are amenable to soft deadlines for their data transfers, such as on-demand media streaming. When the playout buffer is sufficiently full, downloading of data can be paused whenever signal quality degrades, without triggering a disruption in video playback. This reduces wireless energy consumption without hurting user experience.

This paper presents EnVi, a signal-aware scheduler that runs locally on the mobile device, and EnVi Player, an energy-efficient video player that minimizes battery drain during realtime video download. EnVi uses a simple thresholding scheme where the download is triggered whenever the instantaneous signal value goes above a certain signal threshold, and paused when the signal drops below a certain threshold. EnVi self-calibrates the values of these thresholds using local history of signal strength and network throughput that it builds up as the trip proceeds. This approach enables EnVi to adapt to the user's mobility pattern, maximizing energy saving while ensuring that the user's experience is not degraded. A key strength of EnVi is that it is a self-contained approach that does not require any privacy-

CellNet'13, June 25, 2013, Taipei, Taiwan

sensitive knowledge of the user's route or future signal profile. EnVi also does not require assistance form the network and is able to adapt to different signal conditions.

EnVi employs simple practical heuristics to deliver radio energy savings in diverse real world settings with its various constraints and corner cases. Unlike a closely related previous work- Bartendr [8], EnVi does not require any extensive calibration phase to profile signal quality variations for each user traveled route. Thus, EnVi works even when there are detours in previously traveled routes, and more importantly on entirely new routes as well. EnVi is also resilient to variations in the speed of the user's movement, and signal profile changes due to time of day effects, different cell tower associations and new cell tower deployments.

We have implemented EnVi Player on Android and Windows 7 phones. A preliminary evaluation of the prototype conducted on a popular suburban train route produced avg. radio energy savings of 20.7%, going upto as high as 32.5%.

2. RELATED WORK

2.1 Mitigating Radio Tail Overhead

Cellular radios incur high energy drain after each data transfer while transitioning from active state (CELL_DACH) to idle state, which is commonly referred to as the radio tail overhead. By batching multiple short data transfers into fewer longer transfers one can reduce the number of radio tails [2]. Other approaches invoke a network API call called fast-dormancy to cut down the tail energy by forcing the radio to quickly go to a low energy idle state [6]. Studies [5] have shown that many mobile applications such as Youtube do not efficiently use the available network bandwidth, and effectively keep the cellular radio in high power active for long durations of time. Wireless energy savings of up to 80% can be achieved by downloading video chunks at the highest possible rate to build up enough playout buffer, and quickly move to idle state by invoking fast-dormancy. In this paper, the naive approach we use as baseline downloads video stream at full-throttle rate. Thus, energy savings achieved by EnVi are over the best possible download protocol for video streams proposed by previous studies. In addition, EnVi is focused on mobile settings where signal variations are taken into account for downloading each video chunk.

2.2 Reducing Communication Energy

Prior related work regarding reducing radio energy consumption requires knowledge about the user's mobile usage context, while EnVi does not and starts afresh everytime without requiring any information about the user's future route or signal profile. BreadCrumbs [4] maintains a personalized / trained mobility model per user, using past observations of networking conditions associated with GPS locations, and a Wi-Fi AP quality database. Context-for-Wireless [7] also requires training on and maintaining user-mobility and usage-context history to select either Wi-Fi or cellular network for communication based on network condition estimation, except for its 'Hysteretic Estimation' algorithm that works for short data transfer intervals wherein network conditions don't change often, not applicable in our break-free media streaming under mobility' scenario.

Other approaches to saving radio energy include exploiting multiple radio interfaces [7], batching & prefetching [1], and rate adaptation [9]. EnVi's savings are obtained by only exploiting the signal strength variability in the 3G/H+ cellular networks, and can be complemented with these techniques in the form of WiFi-offloading, multiple parallel video-chunk downloads and per video-chunk bitrate optimization.

Of all the previous studies, Bartendr [8] is the closest work to EnVi. Bartendr predicts future signal quality based on data collected during past drives on the same route. Using future signal predictions, Bartendr employs a dynamic programming algorithm to compute the schedule for downloading video chunks within their specified deadlines such that the total energy consumed for data transfer is minimized. Signal prediction assumes that the speed along any given route does not change significantly over multiple runs, and thus Bartendr is not resilient to variations that are typical of user mobility. Moreover, when signal profiles change due to time of day effects or due to new cell tower deployments, the calibration efforts need to be redone. Bartendr is able to schedule downloads only on routes for which signal profile exists, thus does not work on new routes or even when there are minor detours in earlier routes. EnVi on the other hand does not use signal-profiles but recent-history-based heuristics to adapt to changing signal conditions.

2.3 Channel-state based Scheduling

Cellular base-stations use channel state information to perform fine grained scheduling [3] at *ms* granularity for downlink transmissions to active users in a cell. The objective is to maximize efficiency of channel usage and increase aggregate network throughput. EnVi on the other hand is a client initiated communication based on several factors-current channel state, future signal predictions, application deadline, and energy consumption. The scheduler in base-station does not have this level of context about clients, and thus cannot schedule / buffer packets at the granularity of 10s of seconds.

3. DESIGN

We have the following three design goals, motivated from real-world constraints:

- *Unhampered user experience*- the data download strategy should not negatively impact media playback.
- *No server-side modifications*- the solution should be completely local to the device and not require any server-side modifications, for a seamless deployment.
- *Robust*- the solution should work on arbitrary user routes with unknown signal conditions.

3.1 Scheduling Algorithm

The basic idea behind an energy aware scheduling algorithm is to preferentially prefetch data when the signal is good and hold off downloading when the signal quality degrades, so long as there is sufficient data in the playout buffer. In order to do this, there are three key challenges that need to be addressed.

First, given no apriori knowledge of signal conditions, we need to figure out what is a 'good' signal. A simple static high threshold for a good signal (e.g., -60dbm) will not be robust to various provider networks/user routes. Thus, we need a mechanism where threshold for good signals is *dynamically* determined based on history of observed signals. Second, even with dynamic thresholds, future signal conditions may be significantly different from the past. Thus, we

need a way to *adapt* the thresholds as conditions change. Third, cellular radios incur a *tail-energy overhead* [2] every time the radio is turned on/off. Thus, the algorithm has to avoid frequent radio on/off episodes.

The scheduling algorithm is shown in Figure- Algorithm 1. At a high level, we divide the video into multiple chunks, impose deadlines for each chunk download completion based on the video playout rate, and download them independently. A chunk represents the smallest granularity of data transfer unit. The chunk size is determined by the bitrate (5 sec worth of video, in experiments). With each chunk that gets downloaded, the intermediate deadlines are updated depending upon the average recent chunk transfer time, to account for the current network characteristics. A moving recency-time-window (30 sec. in experiments) is maintained to continuously update expected chunk transfer times. We now discuss how the algorithm addresses the aforementioned challenges.

Radio tail overhead. To address the radio tail overhead challenge, we employ three techniques. First, we use fast dormancy [5] to proactively put the radio in a low powered state (IDLE or CELL_PCH) on poorer signals, and bring it back up to a high power transmission state (CELL_DACH) when the signal quality improves. While fast dormancy is more energy efficient than incurring the full radio tail, it still consumes significant energy (roughly 1W x 3.5 s). Thus, we also enforce *min_radio_up_time* (line 4), wherein, once the radio is turned on, we continue to download for atleast this duration (4 x 3.5 seconds, in our experiments), in order to amortize the tail overhead. Finally (not shown in the algorithm), we handle signal spikes by a) using smoothed signal values and b) building in a hysteresis to the algorithm so that even if signal spikes down momentarily, if good signal values have been seen in the recent past, we continue with the next chunk download.

```
1  while chunks remaining do
2  |  rssi = getCurrentRssi()
3  |  thpt = getRecentThpt(rssi)
   |  // schedule next chunk transfer, contingent on
   |     following conditions
4  |  if radio_on and !min_radio_up_time then
5  |  |  download chunk irrespective of rssi;
6  |  else
7  |  |  if rssi in top recent_rssi_p_thresh percentile and
8  |  |  rssi in top overall_rssi_p_thresh percentile and
9  |  |  thpt in top thpt_p_thresh percentile then
10 |  |  |  start downloading chunk;
11 |  |  else
12 |  |  |  if chunk deadline near then
13 |  |  |  |  download chunk irrespective of rssi;
14 |  |  |  |  deadlineEnforcedThreshUpdate(rssi)
15 |  |  |  else
16 |  |  |  |  if radio_on then stop downloading chunks
   |  |  |  |  and invoke fast dormancy;
17 |  |  |  end
18 |  |  end
19 |  end
20 end
```

Algorithm 1: EnVi's Scheduling Algorithm

Dynamic thresholds. In order to determine what is a good signal to download data in a robust manner, we use past history to determine our thresholds dynamically. However, we found that a single dynamic threshold based on recent history is not robust to the varied conditions seen during our experiments. Thus, our algorithm employs a combination of three different thresholds to determine good signal conditions to download data on (lines 7-9). Our thresholds are based on relative percentiles rather than absolute values of the cellular signal (referred to as rssi- received signal strength indicator) and data throughput.

- *recent_rssi_p_thresh*: The current signal must surpass this percentile threshold of rssi values seen in recent history (aforementioned recency-time-window), to be potentially selected for data transfer. This threshold tries to ensure that the data transfer happens on only the best of the signals seen recently.
- *overall_rssi_p_thresh*: The current rssi must surpass this percentile threshold value of rssi values seen in the entire communication history, to be potentially selected for data transfer. We need this parameter in addition to *recent_rssi_p_thresh* to avoid local sub-optimal decisions, i.e. when the signal strengths have generally been poor recently. Thus, even if current rssi is amongst the best of the recent poor lot, it is not a good candidate overall unless it is in the top *overall_rssi_p_thresh* as well. On the other hand, we cannot rely only on *overall_rssi_p_thresh*; *recent_rssi_p_thresh* adapts faster to changing conditions, allowing the use of higher thresholds if recent signal conditions are much better than overall values.
- *thpt_p_thresh*: Recent throughput observed for the current rssi X must surpass this percentile threshold value of throughputs observed for rssi X in the past, to be potentially selected for data transfer. This ensures that even if the current rssi is amongst the very best, but if we have observed lower throughputs recently for this rssi than what has been generally seen, then it is perhaps better to wait for this, possibly transient, phase to subside (e.g, due to transient congestion, interference, hand-off, etc.).

The percentile thresholds are aggressively set to 95% initially, but adapt automatically as follows.

Adaptive thresholds. While the above thresholds are dynamically determined based on past history, the user may encounter a future that is significantly different from the past. When this happens, we need to update the thresholds to reflect this new reality. One feedback mechanism that identifies such a scenario is the deadline-based forced download wherein a chunk is forced to be downloaded, irrespective of signal, because its deadline is close (`deadlineEnforced ThreshUpdate()`, line 14, algorithm 1). In such a case, if a lower rssi is used for the forced download, while a higher rssi X was left unused earlier, we have paid a higher energy cost owing to tighter thresholds for rssi X. But since we maintain history information, we are able to figure out the reason behind not selecting rssi X earlier, and determine the conditions that would have allowed rssi X to be used during the higher rssi readings in the past. We do this by backtracking in time to detect which thresholds to tweak for rssi X amongst the 3 aforementioned ones, and what corresponding percentile values to set the thresholds to. The threshold(s) are restored back to their original percentile value(s) after some chunks have been downloaded on rssi X (in experiments, 2 x number of chunks downloaded on poorer rssi).

3.2 Scheduler in Action

Figure 2 shows a typical experimental run, highlighting how the scheduling algorithm adapts to the varying signal conditions to maintain a break-free energy-efficient media playback. Initially, to buildup a reasonable-sized playout

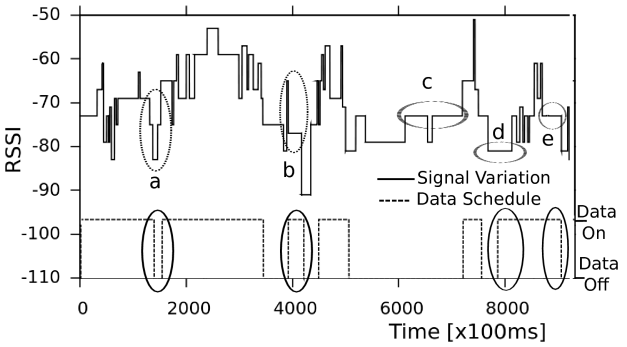

Figure 2: EnVi's data scheduling behaviour

buffer, signal-agnostic data download is initiated. As the session proceeds, when coming across poor signal quality (case 'a' in the figure), the scheduler pauses intermediate chunk downloads, deadline permitting. A signal-agnostic naive scheme would have continued downloading in such a scenario, thereby paying a higher radio energy cost. Upon detecting better signal conditions, the radio is turned back on, but the radio is not immediately turned off if the signal suddenly turns poor – the radio continues to stay on for the *min_radio_up_time* in order to balance the overhead of invoking fast dormancy (case 'b'). As the run proceeds, the scheduler manages to avoid poorer signals (case 'c'), but sometimes chunk deadline constraints may require an immediate download, irrespective of signal, so as to maintain a break-free playback (case 'd'). In this case, data was downloaded at a signal quality poorer than what had been left unused in the past (case 'c'). On detecting this, the scheduler relaxes the selection criteria (i.e., lowering thresholds for what qualifies as a good signal), thereby increasing its probability of selection in the future (case 'e').

4. IMPLEMENTATION

We implemented EnVi on two platforms – Windows Phone 7 (WP7) and Android 2.2. A screenshot of the WP7 EnVi application is shown in Figure 3, where a video from YouTube server is being played along with the recently seen signal values and the status of the download protocol. YouTube server allows us to download a video stream using the highest possible bitrate (without rate throttling) starting at any arbitrary position in the video by specifying the start time relative to the beginning of the video. EnVi automatically determines these parameters for the user requested YouTube video, and encodes it in the HTTP request URL to fetch individual video chunks.

EnVi registers with the OS to obtain updates when the signal strength sensed by the device's telephony component changes. After each chunk gets downloaded, the scheduler uses this current signal value in it's decision regarding whether or not to terminate the current connection and put radio to fast dormancy. Once that happens, the scheduler wakes up every 100ms to read the current signal value and decides whether to initiate a new connection for the next chunk and put the radio back up to its high power state.

For carrying out controlled experiments, we also set up a TCP server that emulated YouTube server, and used EnVi to download video streams of certain length and bitrate as specified in the particular experiment. Instead of performing full video playback each time, we kept the screen on throughout our experiments to take display energy into account. This might affect overall savings but not radio energy component. Any other side effects this introduces equally affects the compared schedulers.

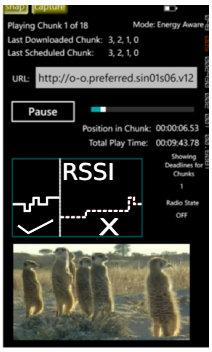

Figure 3: EnVi YouTube player on WP7

5. EVALUATION

We conducted trace-driven simulations and real-world experiments to compare EnVi against other schedulers.

5.1 Simulation experiments

We gathered 51 traces while driving around on multiple routes in two cities with different operator networks – (1) Airtel 3G network in Bangalore (Blr), India and (2) Bell 4G/H+ network in Toronto, Canada. An application on the phone continuously downloaded data from a server on the Internet at full throttle speed during the drive. The trace consisted of per second timestamped samples of instantaneous RSSI (dBm), power consumed (mW) and download throughput (Kbps). The workload for experiments consisted of downloading streams of different lengths for two popular video bitrates – 240p (384 Kbps) and 360p (768 Kbps), and an audio bitrate of 192 Kbps. For comparison, we computed the radio energy savings achieved by four different scheduling algorithms over a Naive scheme that downloads the stream in one go from the time the request is initiated. These algorithms differ in the amount of knowledge that they have regarding both past as well as future information, and thus serve as good reference points for evaluating EnVi.

- **Oracle**: Algorithm that has complete knowledge of both future signal and throughput values; Optimal download schedule is computed using the dynamic programming algorithm described in Bartendr [8]. This provides an upper bound on possible energy savings.
- **Oracle_RSSI**: Algorithm with complete knowledge of only future signal values; throughput values are estimated from historic traces for each RSSI level. Thus, signal prediction is assumed to be perfect in this scenario, but the actual throughput achieved is unknown.
- **Bartendr**: Bartendr algorithm with knowledge of only past traces for each route; this predicts future signal strength values based upon the device's location in the current run. As a result, the future signal and throughput values at all points in the run are only estimates, possibly inaccurate.
- **EnVi**: the proposed local heuristic algorithm that knows only about the recent history in the current run.

5.1.1 *Savings observed with simulations*

Figure 4 shows the energy savings observed over the Naive policy, averaged over 16 traces for Bangalore and 35 traces

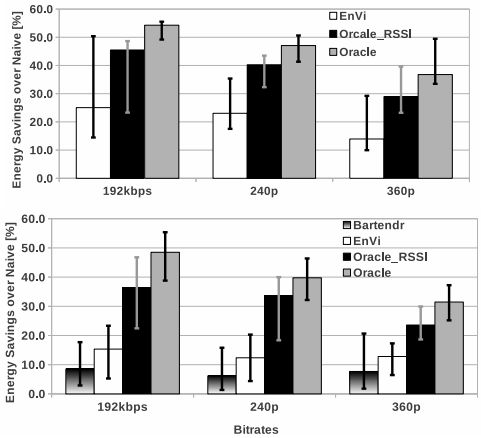

Figure 4: Energy savings over Naive; Above: Blr, Below: Toronto; Stream Length = {10,20} mins

for Toronto. Shown are the 25th percentile, median and 75th percentile values of the savings recorded for each algorithm. We are able to fairly compare with Bartendr for Toronto since we collected traces on an inter-city train with fixed routes and timing schedules. This precise location information is used for Bartendr's training. Table 1 compares the following for Toronto's traces (corresponding table for Bangalore is omitted for space constraints): (i) percentage reduction in radio on time over Naive, (ii) fast dormancy tail counts, and (iii) average increase (improvement) in signal (dBm) during communication as compared to Naive.

As can be seen in figure 4, for Bangalore, EnVi manages to save around 54% of the possible energy savings on average. Comparing with Oracle_RSSI scheme, which assumes a perfect match of signal strengths for the current trip to previous runs, EnVi is able attain 73% of the formers savings. For Toronto, the corresponding numbers are 37% and 47% respectively. Finally, EnVi performs much better than Bartendr when the current trip's signal profile (signal strength vs. location) varies quite a bit in comparison to previously travelled routes because different travel speeds and/or different cell tower associations are seen. It should be noted that if the comparison was made on new routes for which Bartendr did not have any signal profile information beforehand, Bartendr wouldn't have been able to function, while EnVi would have continued to provide similar savings.

As evident from these figures, savings can be further improved (Oracle_RSSI) if accurate future signal profile and route information is available as in public transportation. We are exploring a hybrid algorithm that employs EnVi, when there is no history or when route / signal profile info does not align with the expectation, otherwise switches over to Bartendr that performs well on trained routes.

EnVi is able to save radio energy even when it incurs more tail costs (table 1). Higher tail count is due to the fact that since EnVi has no knowledge about the future signal quality, it has to take scheduling decisions on the fly to balance between the demands of a disruption-free experience and energy-efficient playback. However, by dynamically adapting to changing conditions as the trip proceeds and downloading data on locally optimal conditions, EnVi is able to save on the amount of time the radio stays in a high power transmission state, thereby reducing battery drain.

Comparing energy savings for the two cities, we observe higher savings for Bangalore's traces than for Toronto. This

	192kbps	240p	360p
Radio Uptime Savings			
EnVi	21.7%	18.9%	15.3%
Bartendr	8.7%	13.1%	9.5%
Oracle_RSSI	32.8%	29.4%	21.6%
Oracle	44.7%	38.2%	30.2%
Intermediate Tails			
EnVi	3.7	6.0	9.6
Bartendr	1.1	2.4	3.1
Oracle_RSSI	1.7	3.0	4.4
Oracle	2.2	3.2	4.3
ΔAverage RSSI			
EnVi	3.7	3.3	2.2
Bartendr	1.1	0.9	0.1
Oracle_RSSI	5.1	4.9	2.9
Oracle	5.8	4.5	2.7

Table 1: Improvements over naive, Toronto

	192kbps	240p	360p
ΔAverage RSSI [dBm]			
EnVi	12	10.2	8
Oracle_RSSI	13.5	13.4	10.8
Oracle	12.9	12.4	10.7

Table 2: Improvement in rssi over naive, Blr

is because the proportion of poorer signal strengths seen in Bangalore's traces is much more than Toronto as can be seen in the cdf in figure 5. Thus there is more opportunity to save energy on Bangalore's network. This is also corroborated by comparing the $\Delta AverageRSSI$ metric in Tables 1 and 2.

Also, we find that for lower bitrates energy savings are higher since the scheduler can afford to wait a bit longer while the playout buffer gets consumed at lower rates. This provides more opportunity to find better signal conditions.

Finally, figure 5 also shows that for both cities sufficient energy saving opportunities exist in normal user routes, characterized by a good proportion of poorer signal strengths, to exploit the signal variability for scheduling data transfers on good signals while avoiding the bad ones. This, combined with figure 6's observations (omitting similar trend for Bangalore) that throughputs also vary for each signal value itself along the user route, enable EnVi to be even more selective in scheduling data transfers and thus saving radio communication energy further.

5.2 Real-World Experiments

5.2.1 Setup for live app testing

We compared EnVi and Naive policy during real world experiments using two identical Android phones (Acer S100). We installed EnVi player on one of the phones and signal-agnostic Naive scheme on the other. We conducted multiple runs on a 30 min. long train route, and ran workloads similar to the one in simulation experiments. The two phones start a data stream request at the same time for each run.

5.2.2 Savings observed on live runs

Overall energy consumption for a trip is calculated from the periodic OS-broadcasted 'current battery level' updates that our app subscribes to. Battery consumption, and energy savings, calculated in this fashion includes the energy from all 3 components: (i) radio communication, (ii) screen display, and (iii) cpu processing. To separate out the radio communication component from the overall energy savings, we used average power consumption values measured at different RSSI levels from our earlier experiments (Figure 1) and tail energy cost for fast dormancy. Table 3 shows EnVi's energy savings over the Naive policy averaged over

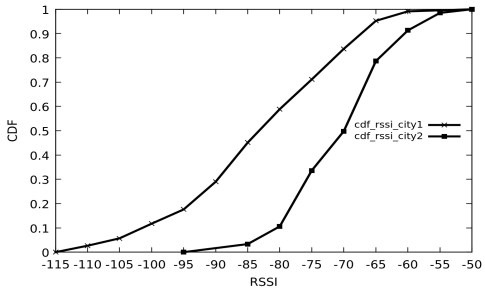

Figure 5: RSSI CDFs for Blr (city1) & Toronto

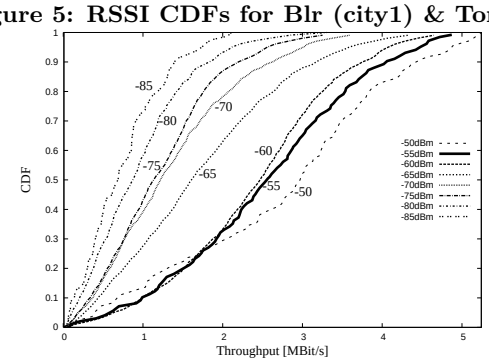

Figure 6: Per-RSSI throughput CDFs, Toronto

18 runs (6 runs each for 3 bitrates; stream length = {10, 20, 30} minutes). Also shown are percentage reduction in radio up time that EnVi achieves by scheduling faster downloads over better signals, the latter being represented as a difference between the average communication signal strength observed with EnVi as compared to Naive.

It is clear that the amount of time the radio stays in high power transmission state (CELL_DACH) is generally lower for EnVi, even after accounting the fast dormancy transition times. This is because although it downloads the same amount of data as the Naive policy does, but it strives to do so on better signal conditions characterized by higher throughputs and hence lower transfer times.

During the live runs, EnVi achieved average savings of 20.7% of radio energy and 10.2% of overall energy compared to Naive, while going upto as high as 32.5% radio energy savings and 21.4% overall energy savings on a few runs. The overall energy consumption metric incorporates all extra processing overheads associated with EnVi's scheduling algorithm, including per-second logging for analysis. It also includes the energy expended while keeping the screen-on for the entire data transfer duration. Also, the OS broadcasted battery level updates are very coarse grained - occur after every 1% drop in the battery level. So, the calculation of energy consumption using these updates alongwith timestamping is slightly less accurate. Additionally, this metric also includes the overhead of emulating fast dormancy on Android [1]. Since there isn't a fast dormancy API on Android yet, we emulated it by enabling/disabling data-connectivity programatically. This has the shortcoming of the device losing its PDP-context and the associated IP address after each call to fast dormancy. When the data connectivity is enabled again, PDP context and connection to the server is re-established resulting in a significant overhead and reduction in our energy savings. We expect to observe better savings with a true fast dormancy facility on device.

[1]In Android context, the term fast dormancy refers to quick transition from H+ to 3G; this differs from our definition of fast dormancy.

	192kbps	240p	360p
Radio Energy Savings	22%	18.5%	18.4%
Radio Uptime Savings	13.7%	13.2%	13%
$\Delta AverageRSSI$ [dBm]	8.4	4.8	4.5

Table 3: EnVi vs. Naive, live runs, Toronto

6. CONCLUSION

In this work we designed EnVi, a light-weight signal-aware scheduler that saves radio energy drain for long standing data transfers by exploiting variation in energy-per-bit values during mobile scenarios. EnVi can adapt to diverse signal variations that exist on different user routes without requiring any pre-calibration, and thus uniquely differs from previous efforts. We have implemented EnVi player on two mobile platforms and shown savings of up to 32.5% of radio energy.

7. FUTURE WORK

Going forward, we plan to perform more comprehensive testing and analysis to judge the efficacy of different thresholds that EnVi employs, attributing percentage gains to each threshold employed. We also want to characterize how EnVi performs with regards to our design goal of a seamless user experience, in cases where the playout buffer runs out of video data owing to possible aggressive scheduling decisions by EnVi. We plan to optimize the EnVi player for video playback and explore the feasibility of optimizing bitrates on a per video-chunk basis depending upon the network characteristics. We are also focusing on augmenting EnVi for background bulk transfers. As discussed in sec-5.1.1, we plan to explore a EnVi-Bartendr hybrid algorithm. Finally, we want to test our conjecture that as a by-product of EnVi's scheduling behaviour, radio communication while at cell edges gets avoided thereby improving aggregate cell throughput and reducing inter-cell interference.

8. REFERENCES

[1] T. Armstrong, O. Trescases, C. Amza, and E. de Lara. Efficient and transparent dynamic content updates for mobile clients. In *MobiSys*, 2006.

[2] N. Balasubramanian, A. Balasubramanian, and A. Venkataramani. Energy Consumption in Mobile Phones: A Measurement Study and Implications for Network Applications. In *IMC*, 2009.

[3] P. Bender, P. Black, M. Grob, R. Padovani, N. Sindhushyana, and A. Viterbi. CDMA/HDR: a bandwidth efficient high speed wireless data service for nomadic users. *Communications Magazine, IEEE*, 2000.

[4] A. J. Nicholson and B. D. Noble. Breadcrumbs: forecasting mobile connectivity. In *MobiCom*, 2008.

[5] F. Qian, Z. Wang, A. Gerber, Z. Mao, S. Sen, and O. Spatscheck. Characterizing Radio Resource Allocation for 3G Networks. In *IMC*, 2010.

[6] F. Qian, Z. Wang, A. Gerber, Z. Mao, S. Sen, and O. Spatscheck. TOP: Tail Optimization Protocol for Cellular Radio Resource Alocation. In *ICNP*, 2010.

[7] A. Rahmati and L. Zhong. Context-for-wireless: context-sensitive energy-efficient wireless data transfer. In *MobiSys*, 2007.

[8] 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 *MobiCom*, 2010.

[9] M. Tamai, T. Sun, K. Yasumoto, N. Shibata, and M. Ito. Energy-aware video streaming with QoS control for portable computing devices. In *NOSSDAV*, 2004.