Adaptive Delay-Tolerant Scheduling for Efficient Cellular and WiFi Usage

Ozlem Bilgir Yetim and Margaret Martonosi
Princeton University
Email:{obilgir,mrm}@princeton.edu

***Abstract*—Today's mobile devices offer multiple network connectivity options with orders of magnitude differences in cost, power, speed and reliability. Given this high variability, dynamic optimization of radio connectivity choice is promising. To increase the flexibility and payoff of such optimizations, we recognize that many applications have significant delay tolerance, which we exploit to schedule data transmissions. This paper proposes and evaluates techniques for cost-optimizing connectivity choice based on application delay tolerance, as well as on predictions of upcoming data usage and connectivity availability. We explore optimal (MILP-based) and heuristic approaches for optimizing this choice while abiding by application performance requirements. Our work studies how errors in predicting data usage or network connectivity impact each approach's success at cost reduction. We evaluate the technique through both simulation and a prototype on an Android smartphone. Overall, our technique averages more than 2X reduction in cellular data usage, and for some scenarios, the reduction is as high as 5X. In addition, the Android prototype also demonstrates the importance of accounting for radio switching overhead and TCP flow migration time.**

I. Introduction

Mobile devices have many connectivity options with widely varying characteristics. For 3G, WiFi and LTE, throughput varies from Mbps to Gbps [4, 20, 26]. Energy consumption per bit varies by more than an order of magnitude: 0.29 uJ/bit for WiFi, 5.86 uJ/bit for 3G, and 0.73 uJ/bit for LTE [12]. In addition, WiFi is often free but cellular usually requires some cost per byte or per month [1]. Sometimes, users might prefer to use free WiFi to avoid depleting their cellular network quota. In others, if WiFi is not fast enough, users might be willing to pay extra for a fast LTE connection for high-throughput or performance-sensitive applications like video-chat.

Although users have choices in networks, the existing strategies for selecting among them are brittle and hard to optimize. In addition, some applications can be tolerant to delays; this allows the possibility of delaying a transmission, in order to exploit some connectivity option that is predicted to soon appear. Currently, however, applications do not provide such delay tolerance information. Moreover, even if delay tolerance information is obtained, it can be difficult to predict connectivity and data usage and plan for unknown future data transmissions under unknown future connectivity options. Furthermore, leveraging connectivity tradeoffs requires the ability to quickly and seamlessly switch between networks, and to sometimes introduce small delays for transmission on delay-tolerant applications in order to use more efficient networks later. These are not well supported in current mobile devices. As a result, users may pay higher per-byte service costs or experience high battery usage because they cannot nimbly employ lower cost or lower energy networks when available.

To address these challenges, this paper explores optimizing connectivity choice as a data transmission scheduling problem. We propose optimal and heuristic scheduling solutions. For each pending data transmission, our approaches choose between using WiFi if available, using cellular, or briefly postponing the transmission if new connectivity is predicted to appear. Our goal is to reduce data usage costs, while abiding by application performance requirements. In addition, we explore whether the optimization accuracy is worth the solution time, or whether heuristic approaches can provide sufficient functionality with lower implementation complexity. Proposed approaches also differ in terms of their reliance on prediction and reducing network energy usage. On an Android smartphone, we show that exploiting application-specified delay tolerance can significantly reduce cellular usage.

The contributions of our work are as follows. First our simulations (with real usage traces as input) show that all of our scheduler approaches decrease cellular data usage significantly compared to the no WiFi baseline. Even the simplest greedy heuristic, which does not consider delay tolerance, cuts cellular data usage by more than 25%. Under moderate delay tolerance, MILP-based can decrease cellular data usage by 80%. Moreover, despite its simple implementation, one heuristic approach ("chunking" (CH)) achieves similar results.

Second, we show that correct prediction of WiFi and data characteristics enables better resource utilization and better scheduling. Scheduler approaches that rely heavily on prediction, however, are less resilient to dynamic or hard-to-predict situations. Because our MILP-based approach relies the most on advanced planning, it is the least resilient to prediction errors. In contrast, heuristic techniques using little or no prediction perform more resiliently across different scenarios.

Third, even though the greedy heuristic is the most error resilient, it uses over 100× more scanning energy than the other approaches even under high prediction errors. When prediction errors occur, CH is the winner if both cellular data usage and network energy usage are considered.

Fourth, we also evaluate the cellular/WiFi optimizing techniques in an Android smartphone using delaying and seamless network switch mechanisms. Our approach is novel in accounting for system delay when switching networks and migrating flows. This saves roughly 20% in cellular data usage.

Fifth, we explore application delay tolerance by adding support for *pausing* data transmissions, in addition to switching between connectivity options. With this support, our Android smartphone prototype reduced cellular usage by 45%.

II. Optimizing Network Selection

We frame network connectivity selection as a scheduling problem with three options: transmitting data via cellular network, transmitting via WiFi network, or pausing data transmission to wait for WiFi by exploiting application delay tolerance. The overall problem scenario considers a person using a mobile device as they move through their day. Sometimes WiFi is available, sometimes not. For each data transmission, if WiFi is not connected, the question is whether to send it now, or wait for WiFi that might appear soon. Constant WiFi scanning

Material based upon work supported by NSF Grant CNS-1135953.

978-1-4799-4786-7/14/$31.00 ©2014 IEEE

TABLE I. COMPARISON OF PRIOR AND PROPOSED SCHEDULING METHODS.

	Offloading Method	Cellular vs WiFi Decision	Data Prediction?	Data Delay?	Data Chunking?	WiFi Prediction?	WiFi Scanning?	Commodity Phone?
Proposed	Greedy (GH)	No scheduling, use WiFi if available	No	No	Yes, use WiFi if available, then switch	No	Every N seconds	Yes
	Non-Chunking (NCH)	Use WiFi now if predicted available, schedule for future if predicted soon	No	Yes	No, use WiFi only if long enough for full data request	Yes (avail. and throughput)	Only when a transmission is scheduled	Yes
	Chunking (CH)	Same as above	No	Yes	Yes, greedily use shorter WiFi windows for data chunks	Same as above	Same as above	Yes
	MILP-based (MILP)	Optimal schedule using MILP, use CH if MILP infeasible	Yes	Yes	Yes, optimal chunks considering predicted data requests	Same as above	Same as above	Yes
Prior	Wiffler [6]	Use WiFi if available, delay if WiFi predicted soon	No	Yes	Yes, use WiFi is available, then switch	Yes, only throughput	Every 1 second to check WiFi avail. if data is waiting	No, requires proxy support
	Rahmati and Zhong [21]	No scheduling, try WiFi connection and use it if overall energy is better	No	No	No, use WiFi if predicted to be available	Yes, only availability	Scan if probabilistically finding and using WiFi is more energy efficient than using current network	No, modified application makes the selection

$$Total\ Cellular\ Data\ Usage: \sum_{s\in S, u\in U_s} (US_u + H_u) \times ns_{u,M} \quad (1)$$

$$Availability\ \ Constraint: \ \forall i \in S, u \in U_s : \tau_u \geq A_s \quad (2)$$

$$Deadline\ \ Constraints: \ \forall s \in S, u \in U_s, n \in N : \tau_u + (US_u + H_u)/BW_n + L_n \leq T_{max} \times (1 - ns_{u,n}) + E_s \quad (3)$$

$$Sequencing\ \ Constraints: \ \forall s \in S, u \in U_s, u' \in U_s, u' \leq u : seq_{u,u'} = 1 \quad (4)$$

$$\forall s \in S, k \in S, u \in U_s, u' \in U_k, u \neq u' : seq_{u,u'} = 1 - seq_{u',u} \quad (5)$$

$$Networks\ \ Constraints: \ \forall s \in S, u \in U_s : \sum_{n\in N} ns_{u,n} = 1 \quad (6)$$

$$\forall s \in S, u \in U_s, n \in N, n \neq M : ns_{u,n} \times (B_n + O_{WiFi}) \leq \tau_u \quad (7)$$

$$\forall s \in S, u \in U_s, n \in N : \tau_u + (US_u + H_u)/BW_n \leq T_{max} \times (1 - ns_{u,n}) + F_n \quad (8)$$

$$Bandwidth\ \ Constraints: \ \forall s \in S, k \in S, u \in U_s, u' \in U_k, u \neq u' : \tau_{u'} + (US_{u'} + H_{u'})/BW_M + O_{WiFi} \leq \tau_u + T_{max} \times (2 - ns_{u',M} - seq_{u,u'} + ns_{u,M}) \quad (9)$$

$$\forall s \in S, k \in S, u \in U_s, n \in N, n \neq M, u' \in U_k, u \neq u' : \tau_{u'} + (US_{u'} + H_{u'})/BW_n + O_{Cell} \leq \tau_u + T_{max} \times (3 - ns_{u',n} - ns_{u,M} - seq_{u,u'}) \quad (10)$$

Fig. 1. MILP formulation to optimize cellular data usage.

TABLE II. FORMULATION VARIABLES, CONSTANTS AND SETS USED IN THE MILP FORMULATION GIVEN IN FIGURE 1.

Term	Description	Type
A_s	Availability time of data stream s	Constant
B_n	Beginning time of network n	Constant
BW_n	Bandwidth of network n	Constant
E_s	Deadline to receive/send data stream s	Constant
F_n	Ending time of network n	Constant
H_u	Header size of sched. unit $u \in U_s$	Constant
L_n	Latency of network n	Constant
M	Cellular network index	Constant
N	Networks	Set
$O_{WiFi/Cell}$	System overhead for switching to new interface and migrating the flows	Constant
S	Data streams	Set
T_{max}	Time upper-bound	Constant
U_s	Sched. units belonging data stream s	Set
US_u	Payload size of sched. unit $u \in U_s$	Constant
$ns_{u,n}$	Schedul. unit $u \in U_s$ is scheduled to network n	Var $Binary$
$seq_{u,u'}$	Sched. unit $u \in U_s$ is scheduled after sched. unit $u' \in U_k$ ($u \neq u'$, $A_s < E_k$,$A_k < E_s$)	Var $Binary$
τ_u	Schedule time of sched. unit $u \in U_s$	Var $Real$

is energy intensive, but always using cellular can be costly. It is useful to be able to predict and schedule upcoming data requests and to predict the WiFi windows that may be used to transmit them. However, if scheduling decisions rely heavily on predictions, the effect of prediction errors on these decisions is also important. These prediction effects and scheduling decision questions are the focus of this paper.

Deciding which network to use and when to delay transmissions depends on *network characteristics*, *application characteristics*, *user objectives*, and *system overheads*. *Network characteristics* include transmission energy, maximum throughput, cost, and availability. *Application characteristics* include the size and arrival times of data to be transmitted, and its delay tolerance. *User objectives* vary from person to person and from situation to situation. Some examples of user objectives include maximizing performance, minimizing battery usage, and curtailing cellular network usage.

System overheads are inevitable in real implementations and yet often ignored in prior work. The primary overheads are the time required for switching between cellular and WiFi channels and migrating data flows. Switching to WiFi takes longer than switching to cellular because of extra steps required, such as scanning through multiple access points, picking one to associate with, and often requesting an IP address with DHCP [18]. In addition to switching, the flow migration mechanism used for seamless network switches brings additional overhead. We account for both.

Scheduling Approaches: Table I compares our approaches with each other and with two previously-proposed network adaptation methods. Wiffler is a simple and effective approach, but requires high scanning energy; it checks WiFi every second if there is data waiting [6]. The paper also does not discuss implementation of the required proxy support for interface management both in server and client side. Rahmati and Zhong studied network condition prediction from context information for energy efficient data offloading [21]. However, they do not focus on scheduling mechanisms to exploit application delay tolerance, as we do.

We evaluate four scheduling approaches, *greedy heuristic (GH), non-chunking heuristic (NCH), chunking heuristic (CH), and MILP-based (MILP)*. These approaches decide on whether to use cellular or WiFi network, when to start the transmission, and how much data is to be transmitted. *GH* is a simple baseline which reduces cellular usage by greedily connecting to WiFi if possible. This approach works similarly to on-the-spot offloading already supported on current phones [15]. However, that model relies on application support for reconnections in case of network changes. Instead, our system design seamlessly migrates network flows without application support [19], and exploits delay tolerance by pausing and resuming TCP connections.

The *NCH* and *CH* approaches make scheduling decisions for each current data stream by predicting future network availability and throughput. If the capacity of predicted WiFi is insufficient, they schedule the data to cellular network. While NCH fits data into available WiFi windows based only on the full size of the request, CH chunks data into smaller scheduling units to exploit shorter WiFi regions. As a result, CH is better able to reduce cellular data usage and hence decrease transfer energy, but might have increased scanning energy costs since it can schedule the scheduling units to more WiFi regions. (In MILP, CH, and NCH, WiFi duration is compared against the switching time overhead. If the predicted WiFi duration is too small to accommodate the switching time overhead, these approaches do not consider connecting to it.)

The *MILP* approach uses Mixed Integer Linear Programming (MILP) to find the minimum-cost schedules by breaking data streams with varying size and delay tolerance into smaller *scheduling units*. In [7], we proposed an optimal scheduling framework using MILP and briefly experimented with it under different data and network scenarios. However, that scheduler does not account for system overheads, and it allows multiple physical interfaces to be ON at the same time (not commonly supported in real systems). We altered these aspects here to be applicable to the real system Android implementation.

Figure 1 and Table II shows the MILP formulation and its parameters. This formulation minimizes Eq. 1, the sum of bytes transmitted through cellular interface. Eq. 2 ensures that data is scheduled after it is generated and Eq. 3 ensures that all transmissions finish before the application deadline. Eq. 4 and 5 are constraints to ensure the correct ordering of scheduling units. Eq. 6 guarantees that each scheduling unit is scheduled to one and only one network. Eq. 7 accounts for system overheads and ensures that transmissions start on WiFi only after the WiFi network becomes available. Similarly, Eq. 8 guarantees that transmissions finish before a network becomes unavailable. Eq. 9 and 10 ensure that overhead is considered while changing networks if two consecutive scheduling units are scheduled to different networks. For the problem sizes in our experiments, this optimization can be solved in less than ten minutes using CPLEX on a 2.33 GHZ Intel Xeon Quad-Core CPU with 16GB memory. On the other hand, heuristic approaches can be solved in linear time. One goal of this work is to see when MILP's extra complexity is worthwhile.

Responding to Prediction Errors: If data or network prediction errors occur, the scheduler's plans for upcoming transmissions may not hold. For example, WiFi might not be available at the time the data is scheduled, the estimated data size might be smaller or larger than the real data size, or from the application, data might arrive earlier or later than the estimated arrival time. The following rules address these situations. If estimated data size is bigger than the real size, excess data is sent with the last scheduled unit. If there are scheduling decisions for data that has not arrived yet (i.e., data arrival is later than the estimated time), those scheduling decisions are skipped. If data eventually arrives but all decisions about it have already been skipped, the data is transmitted right away. These first three rules apply for MILP since it is the only one which makes decisions based on data predictions. Finally, if data is scheduled to a network which does not appear at the predicted time, our retry interval, t, is decided by $t + D_{size}/BW_{WiFi} <= D_{size}/BW_{Cell}$. The interval is set conservatively so that at decision time, waiting and transmitting via WiFi will never take longer than transmitting right away with the cellular network.

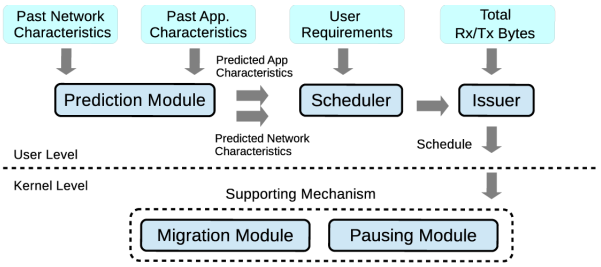


Fig. 2. Design diagram.

III. System Implementation

Figure 2 shows our system design with five major components: Prediction Module, Scheduler, Issuer, Migration Module, and Pausing Module. The prediction module, scheduler and issuer operate in user level and use the underlying kernel-level pausing and migration mechanisms. We have implemented our CH and NCH schedulers on an HTC G2 smartphone which runs on Android version 2.3.7 on kernel version 2.6.35, but the modules are easily transferable to newer versions of Android.

Migration Module and Delay Tolerance: Migrating flows among different interfaces is a key part of our design. Since IP addresses change when a new network connection occurs, seamless connection over different networks is problematic. For our migration module, we utilize the flow migration of Serval [19] so that the transmissions can be migrated from one interface to another seamlessly. Serval proposes a new layer to the network stack which uses service IDs to identify services instead of network address and port number, and flow IDs to identify the ongoing transmissions. Thus, network address changes do not disrupt ongoing services.

We modified Serval to support delay tolerance. We encode a service's delay tolerance using Serval IDs. Service IDs in Serval consists of as many as 256 bits, without any overhead for an active connection since service IDs are in use only during the connection establishment. In our system, lower bits of the Serval service IDs can be used to indicate the delay tolerance of the service. For example, an email service may be offered with through service IDs 0-15 with increasing delay tolerance. With this usage, every "write" call to the socket of a service defines a new data package that must be delivered within the delay tolerance indicated by the service.

Pausing Module: This module pauses the flow of data when delaying a transfer is warranted by the Issuer. We focus on TCP connections here. TCP has congestion-control and flow-management techniques that use several timers to implement window management. However, in order to utilize the delay tolerance of applications, the TCP layer also needs to be able to temporarily pause a transfer and resume it later (same network or elsewhere) when unpause is called.

We implement pausing by changing the management of the data send queue in the TCP layer. TCP first breaks application data into segments and forms a data queue. It then sends a window of data to the receiver over the network and waits for acknowledgments before sending the rest of the segments. In our module, whenever a pause is requested, the queue management pauses. Packets that are currently in flight are handled by the TCP layer just as if the flow had not been paused at all. When the pausing module continues the transfer, the queue is unpaused, and the rest of the packets in the queue are pushed through the network as they would have been. This method is advantageous because the application's only responsibility is to tell the system a flow's delay tolerance.

Predictor, Scheduler and Issuer Modules: The prediction module makes predictions about future network or data behavior. We assume a simple history-based predictor using a sliding window average. More advanced predictor methods [10, 17] can be used here, but this work focuses exploring different schedulers. We do, however, experiment to see how their different prediction reliance affects results.

The Scheduler operates according to one of the approaches previously described, and the Issuer is responsible for scheduling the data to the networks, as stated in the schedule output. It has five main responsibilities: (i) tracking the bytes transmitted for each application data since the scheduled unit size varies depending on the scheduling approach, (ii) initiating pausing for the applications that need to wait for a future network or that gave priority to another application with a shorter deadline for the current network, (iii) turning on/off the interfaces before switching networks, (iv) initiating migration of flows between networks, and (v) deciding how to adjust the plan in case of prediction errors following the rules explained in Section II.

IV. Experimental Methodology

We use both simulations and real implementation to evaluate our scheduler approaches. For simulations, we implemented our schedulers in Python. MILP is written in AMPL and IBM CPLEX 12.05 is used to solve it [3, 13]. We repeat each simulation 10 times since some of our input parameters are chosen from probabilistic distributions.

Application Data Usage: We used both real and synthetic workloads with varying averages in our experiments. Our real data usage traces were logged from different users using an Android app we built. It collects total transferred bytes by gmail, google docs and gallery application (when auto back up is set) per-second. We logged 5 days of usage for 7 individuals who were Android Ice Cream users and who were already using these applications regularly.

For our real system implementation, we used synthetic data generated by our client-server file transfer application on Android. This application takes the size and arrival times of data as input, and sends corresponding data to our local server at designated times. Data sizes and inter-arrival times vary depending on the application. For example, the average Web page size is almost 700 KB [25] and file transfer data sizes can be a few MBs. Moreover, data arrival intervals can be tens of seconds up to few hours [15, 21]. For our file transfer application, we used exponentially distributed data with size averaging 4MB and inter-arrival averaging 200s.

Although real-time audio/video streaming applications are very sensitive to transmission delays, gmail, google docs or photo back up in gallery application can tolerate delays up to a few hours. Despite this, applications like gmail often transmit the data immediately from whichever network is available. On the other hand, applications like gallery always wait for and use only WiFi (if auto back up feature is set). Instead of not waiting for WiFi at all or waiting for WiFi forever, we study the effect of different fixed values of delay tolerance on these applications. We vary this tolerance from 50 to 400s. More adaptive or dynamic techniques for deducing or predicting delay tolerance would be interesting for future work.

Network Connectivity Options: Previous work has shown that the average WiFi inter-connection time and duration can be a few tens of seconds to a few hours, depending on the distribution of WiFi and user's mobility [8, 15]. Thus, to generate similar WiFi scenarios, we vary the WiFi inter-connection time between 25s and 400s and WiFi duration from 20s to 320s. In our real experiments, the phone is placed in a WiFi and T-Mobile 4G range. We measured the average WiFi upload bandwidth as 6.64 Mbps and average cellular upload bandwidth as 800 Kbps for the observed time.

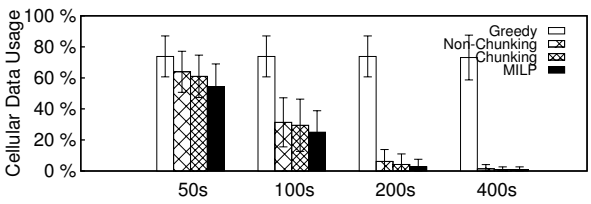


Fig. 3. Leverage of MILP, NCH, and CH increases with increasing delay tolerance of gmail.

System Overheads: System overheads depend on the hardware, software and network carrier. We measure the time between the start of an interface switch and the moment the new connection becomes available on the Android smartphone. Our measurements show that it takes around 1.2s to switch from cellular to WiFi network, and it takes around 0.5s to switch from WiFi to cellular network. Moreover, our experiments show that necessary service table updates can take between 4s and 13s and migrations take around 0.5s. Thus, for the simulations, we use 5s for the switching and migration overhead from WiFi to cellular, and 15s from cellular to WiFi.

Introducing Prediction Errors: We focus on four parameters that are predicted by one or more of the schedulers: WiFi duration, WiFi arrival time, data size and data arrival time. For the simulations, rather than focusing on a single prediction scheme and error rate, we consider a range of possible error rates that might arise from different prediction approaches. For each parameter, we randomly choose and apply an error value from a uniform distribution. For example, for 100% error, we sample an error value from a uniform distribution ranging from -100% to 100% and modify the original value by this amount of error. For real-system experiments, history-based predictor we built introduces errors inherently.

Energy Usage: We compare our scheduling methods in terms of resulting scanning and transfer energy usage. For scanning energy calculations, we take scanning time as 1.1s and scanning power as 0.56 W [20]. For WiFi mispredictions, the Issuer triggers scanning for an extra t time as explained in III. For the transfer energy, we use 0.29 uJ/bit for WiFi uploads and 5.86 uJ/bit for 3G uploads [12].

V. Results

A. Comparison of Scheduler Approaches

Here, we compare cellular usages of the schedulers under perfect data and WiFi prediction. Our data traces are for five days, thus we show the resulting five-day averages and standard deviations of different days in the graphs.

Effect of Delay Tolerance: Figure 3 shows the cellular data usages under varying delay tolerance values for a single user's gmail trace for an average WiFi duration of 20s and inter-connection time of 25s. Lower bars indicate lower cellular data usage. For high delay tolerance, the more successful scheduling techniques almost reach 0 cellular usage; almost all transmissions are scheduled onto WiFi. Overall, the graph clearly shows that all techniques offer significant improvements over cellular-only—even the simplest GH beats it by roughly 25%. GH does not use delay tolerance in its policy, so its cellular usage remains unchanged across different delay tolerance settings. For the other three techniques, increasing delay tolerance increases their ability to wait for and exploit future WiFi opportunities. At 100s delay tolerance, MILP improves over the cellular-only baseline by roughly 75%.

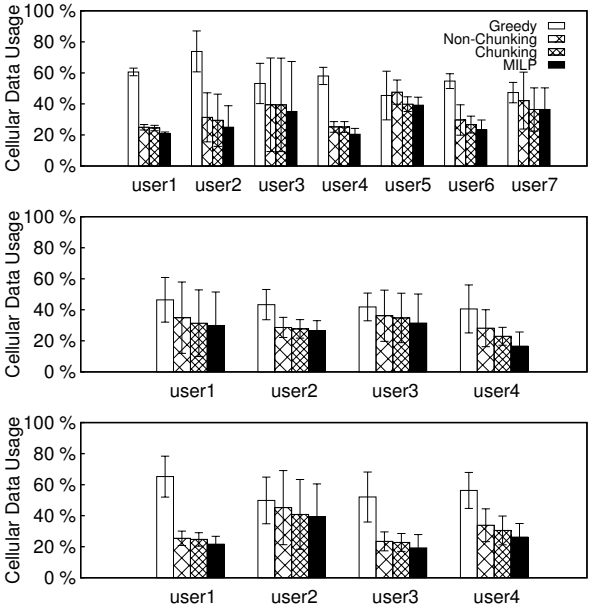
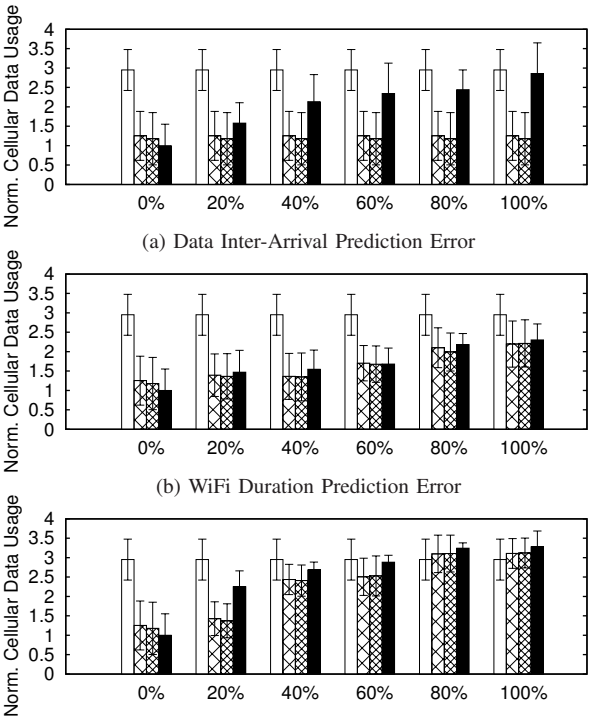


Fig. 4. Cellular bytes usage values for given application (some users do not use all of the applications) with the schedulers. Upper: Gmail, middle: Google docs, lower: Gallery.



(a) Data Inter-Arrival Prediction Error

(b) WiFi Duration Prediction Error

(c) WiFi Inter-Connection Prediction Error

Fig. 5. Overall, NCH and CH are better than MILP under high errors.

MILP gives the minimum cellular data usages at all delay tolerances. At high delay tolerances, both CH and NCH performances improve and achieve similar result as MILP. Both are able to achieve almost 0 cellular data usage with sufficient delay tolerance. CH typically slightly outperforms NCH because it is able to break data into smaller chunks to exploit shorter periods of WiFi. For example, at 50s delay tolerance, CH can achieve 4% less cellular data usage. This behavior becomes more prominent when the data sizes increase. It is advantageous in terms of using WiFi more effectively, but

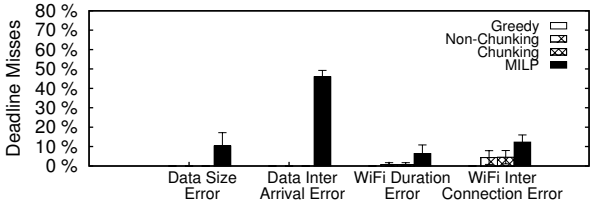


Fig. 6. Deadline misses for different schedulers under high error.

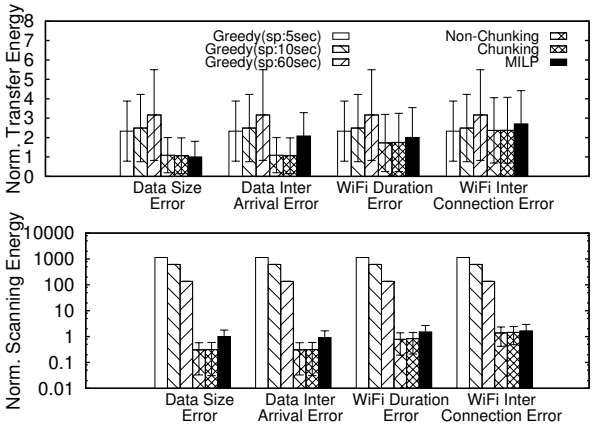


Fig. 7. Energy under high prediction error. Upper: Transfer energy, lower: Scanning energy.

results in higher number of interface switches, hence results in more WiFi scanning energy. Section V-B compares the techniques on energy usage.

Effect of WiFi Conditions: WiFi duration and inter-connection time (time between WiFi availability windows) also affect cellular data usage. With longer WiFi durations, cellular usage decreases with all schedulers. Space constraints preclude a fuller elaboration of these results.

Effect of Application Usage: Figure 4 shows the cellular data usages for gmail, google docs and gallery application for an average WiFi duration of 20s and inter-connection time of 25s. Application delay tolerances are set to 100s. There are significant savings in cellular data usage for all three application types. Leverage compared to GH varies depending on the user and the application type. MILP can achieve 10-60% less cellular data usages compare to GH. Even though NCH performs almost as well as CH, it is up to 8% worse on cellular usage for some users. Moreover, the standard deviations show that CH and NCH can exceed GH for some users. (When new data arrives close to end of WiFi connectivity, one cannot switch on WiFi fast enough to use it before its connectivity duration is over. GH opportunistically turns on WiFi whenever it appears, and MILP pre-plans the switching when it predicts a data arrival, so they do not experience this same issue.)

Overall, assuming perfect prediction of data usage and network conditions, MILP performs well as expected. In addition, CH can be nearly as successful in finding the optimal schedule and minimizing the cellular usage. Next, we explore these tradeoffs when prediction errors are introduced.

B. Effect of Prediction Errors

Effects on Data Usage: Figure 5 shows normalized cellular data usages when there are prediction errors related to data arrival time, WiFi duration and the inter-connection time. (Errors in predicting data sizes do not substantially affect cellular data usage.) These are for the default experimental values of gmail data usage trace, 100s delay tolerance, 20s average

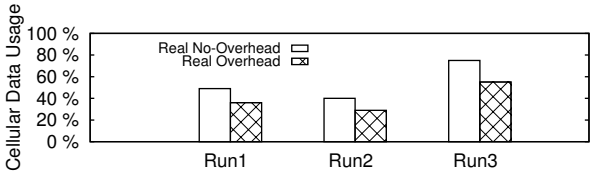


Fig. 8. Effect of taking overheads into account in real system design.

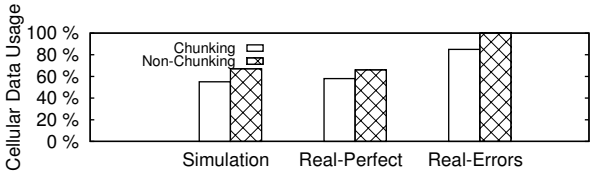


Fig. 9. Cellular data usage with the proposed design in real system design.

WiFi duration and 25s average inter-connection. Graphs are normalized to MILP under no prediction errors.

First, since GH operates without *any predictions* on either WiFi or data, it is unaffected by variations in predictability. Second, CH and NCH use predictions about WiFi availability, but not about data; they only consider data already in their queue. As a result, they are susceptible to WiFi prediction errors, but not data prediction errors. Since MILP makes predictions about both data and WiFi, it is susceptible to all four types of error.

Figure 5a show MILP's sensitivity to data arrival errors. Because it seeks to aggressively exploit WiFi, based on data transmission size predictions, it must revert to cellular usage when more data arrives than expected, or when data arrives earlier/later than expected and WiFi is unavailable. For these reasons, heuristic approaches (which schedule only after data has arrived) can outperform MILP. For example, CH achieves $3\times$ better cellular data usage than MILP when there is high prediction error on data arrival time. Our history-based data arrival time predictor has variable performance for different users and applications. In the best case, it achieves 26% standard deviation in predicting next data arrival. Figure 5a shows that even with 20% prediction error (14% standard deviation) MILP is almost 50% worse than CH. For MILP to be effective, in addition to additional resources used for the solver, more advanced data arrival predictors are necessary.

Three of the four techniques are potentially susceptible to WiFi prediction errors, with MILP affected more than CH and NCH. Even though it finds better schedules under perfect prediction, MILP's cellular data usage becomes worse when errors are introduced. At high WiFi inter-connection time prediction error (Figure 5c) MILP's cellular data usage increases by more than $3\times$ and becomes even worse than the simple GH. CH and NCH results in similar cellular data usages when prediction error is high for this data trace. However, when data sizes are bigger, NCH still achieves better cellular data usage than CH since NCH only schedules data in its full units. Section V-C validates this in real system experiments.

Effects on Energy Usage: Figure 7 shows the relative scanning and transfer energies under 100% prediction errors for the gmail data trace. All schedulers except GH use WiFi predictions to make scheduling decisions, so they do not need constant WiFi scanning. When MILP, CH, or NCH are used, the issuer scans and establishes a WiFi connection only if there is scheduled unit that period. On the other hand, GH scans WiFi periodically whenever a WiFi connection is not already established. As a result, GH scanning energy is not affected by errors. Considering GH schedulers with 5s, 10s, and 60s scanning periods, we see that as scanning becomes less frequent (from ever 5s to 60s) scanning energy decreases almost $10\times$. Compared to NCH and CH, however, GH scanning energy is still more than $100\times$ higher even under 100% WiFi inter-connection and duration prediction error. When there is 100% prediction error on data size and arrival time, MILP results in $3\times$ more scanning energy than CH and NCH. Overall, MILP uses more WiFi regions than CH, and CH uses more WiFi regions than NCH, so MILP's scanning energy is the highest, with CH much lower and NCH slightly lower still.

When cellular transmit energy per byte is higher than WiFi transmit energy, transfer energy increases with cellular data usage. Although infrequent scanning reduces scanning energy, it results in missing more WiFi regions, and thus larger transfer energy. Although MILP experiences prediction errors in data size and arrival time, GH with 60s scanning period uses $3\times$ more transfer energy than MILP. Even with high WiFi inter-connection prediction error, the transfer energy of MILP, CH, or NCH is almost $1.3\times$ less than GH (60s period). Considering both scanning and transfer energy, MILP, CH, and NCH all clearly outperform GH—even under high mispredictions.

Performance Effect: In addition to the potential impact on cellular data usage, another issue pertains to how prediction errors affect whether data transmissions abide by their deadlines. (That is, when an application specifies a delay tolerance preference, it should not be exceeded.) Figure 6 shows the percentage of data transmissions that finish after the deadline. Due to prediction errors, the MILP scheduler misses the deadlines more often. This is because the MILP technique is most aggressive in trying to optimize for using different WiFi windows, so variations on the size/arrival of either data or WiFi can affect its timeliness the most. The other heuristics miss fewer deadlines, largely because they are less aggressive and orchestrated in their attempts to use small windows of WiFi availability. For example, NCH uses much less WiFi, and when there are high prediction errors regarding WiFi duration, it misses roughly $2\times$ fewer deadlines.

C. Real-System Measurements

We have implemented Android prototypes of Section III's design. In real systems, the time to switch between interfaces can be comparable to WiFi availability durations, and thus these overheads become important for the scheduler's decision making. We test our approach using a file transfer application sending data of exponentially distributed data sizes with an average size of 4MB and arrival time of 200s. We run the prototype with network characteristics that match three of the scenarios previously explored in simulations.

Effect of System Overheads: Figure 8 shows the achieved cellular data usage in our CH implementation, with and without taking system overheads into account. When the scheduler does not account for switching overheads, it is optimistic in its estimate of WiFi capacity. The issuer then needs to send some of the excess data through the cellular network instead. If the scheduler accounts for switching overheads, it can better plan for them. Figure 8 shows this results in better cost savings—the difference can be up to 20% in these examples.

Effect of Prediction Errors: Figure 9 shows the cellular data usage CH and NCH. We can achieve results similar to simulations in a real system with perfect prediction. (The small difference is due to varying network bandwidths in the real implementation.) CH can save almost 45% cellular usage. To predict the next WiFi duration and arrival time, we use a history-based predictor, averaging the past times across a large window. This basic predictor inherently introduces prediction errors. Under high prediction errors, both approaches have an increase on the cellular data usage. CH's cellular data usage

increases by 25% and NCH cellular data usage increases by almost 35% and reaches to baseline. Overall, even though accurate WiFi prediction is undeniably important, cellular data usage reduction is still achievable with imperfect predictors.

D. Summary of Results

CH, NCH and MILP all improve greatly over the simple baseline GH; this indicates the promise for exploiting application delay tolerance. Our most sophisticated technique, MILP, works well when there are no errors in predicting the data arrival and connectivity characteristics it relies on. Given MILP's complexity, we sought out simpler alternatives and devised a much simpler heuristic method, CH, that achieves similar cost savings as MILP, but with better real-world potential due to its reduced reliance on context and data prediction. CH and NCH behave similarly for small transmit sizes, but for the larger data sizes seen in our real system prototyping, CH performs substantially better than NCH.

VI. Related Work

Using different wireless channels: Different forms of network discovery and selection problems have been posed and studied before [2, 5, 6, 15] but are incomplete. Some prior work does not study constraints of mobile phones, others rely heavily on energy-intensive WiFi scanning, others do not exploit application delay tolerance, and others consider system alternatives but do not tackle the scheduler problem itself. Our previous work focused on application delay tolerance and scheduling [7], but this current work is much more complete in terms of realistic design and detailed technique comparisons.

Power/performance effects of using different wireless channels have also been studied [21, 22]. Again, the prior work either does not exploit delay tolerance or makes hardware assumptions incompatible with off-the-shelf phones. However, our work runs on commodity phones and uses delay tolerance to increase network selection's energy-saving potential.

Seamless switching between wireless channels: Several proposals exist for seamless switching among different wireless channels. These include new transport-layer design [11], changes in disconnection sequence [18], adopting mobile IPv4 mobility protocol [27], and adding a new layer on top of existing IP networks [19]. In order to support applications keeping their connection during network switches, we used the Serval stack [19] which uses *service id*s and *flow id*s to identify services and maintain ongoing data transmissions even if destinations move or disconnect.

Network scheduling: To manage network resources, increase bandwidth utilization, solve network congestion and guarantee certain QoS levels for different services, network scheduling has been studied widely in the literature, though only marginally related to our work on optimizing connectivity choice. Some work proposes scheduling frameworks which maximize network throughput [9, 16], others focus on QoS and develop methods for minimizing end-to-end delays or router delays [24, 28], and others explore efficient DTN routing [14, 23]. None of these evaluate scheduling approaches for cellular/WiFi network selection as we do.

VII. Conclusion

This paper optimizes network connectivity while abiding by user cost preferences and application delay tolerances. We evaluated four different scheduling approaches, greedy, non-chunking, chunking and MILP-based, to understand their relative effectiveness in minimizing the cellular data usage. All four are very promising improvements over today's state-of-the-art. When data and network prediction can be performed accurately, MILP-based approach offer the largest savings in cost and energy. For less well-predicted scenarios, a chunking heuristic (CH) offers the best approach, with good performance that is more resilient to dynamic changes in data or network behavior. By implementing CH on an Android smartphone, we confirmed its overall effectiveness, and also demonstrated the importance of accounting for switching overheads when building connectivity schedulers of this type. Overall, our work offers insights by prototyping real connectivity optimizers, and considering a range of design alternatives.

References

[1] AT&T 5GB/month data plan cost. [Online]. Available: http://www.att.com/shop/wireless/data-plans.html,2012

[2] "A 3G/LTE Wi-Fi offload framework: Connectivity engine (CnE) to manage inter-system radio connections and applications," Qualcomm Incorporated White Paper, 2011.

[3] AMPL: A modeling language for mathematical programming. [Online]. Available: http://www.ampl.com/

[4] G. Ananthanarayanan and I. Stoica, "Blue-fi: enhancing wi-fi performance using bluetooth signals," ACM MobiSys, 2009.

[5] J. Arkko *et al.*, "Network discovery and selection problem," RFC 5113, 2008.

[6] A. Balasubramanian *et al.*, "Augmenting mobile 3G using WiFi," ACM MobiSys, 2010.

[7] O. Bilgir Yetim and M. Martonosi, "Adaptive usage of cellular and WiFi bandwidth: An optimal scheduling formulation," in *Proc. ACM International Workshop on Challenged Networks, (CHANTS)*, 2012.

[8] V. Bychkovsky *et al.*, "A measurement study of vehicular internet access using in situ Wi-Fi networks," ACM MobiCom, 2006.

[9] P. Chaporkar and A. Proutiere, "Adaptive network coding and scheduling for maximizing throughput in wireless networks," ACM MobiCom, 2007.

[10] P. Deshpande *et al.*, "Predictive methods for improved vehicular WiFi access," ACM MobiSys, 2009.

[11] X. Hou *et al.*, "Moving bits from 3G to metro-scale wifi for vehicular network access: An integrated transport layer solution," IEEE ICNP, 2011.

[12] J. Huang *et al.*, "A close examination of performance and power characteristics of 4G LTE networks," ACM MobiSys, 2012.

[13] IBM ILOG ODM enterprise developer edition. [Online]. Available: http://pic.dhe.ibm.com/infocenter/odmeinfo/v3r4/index.jsp,2010

[14] S. Jain *et al.*, "Routing in a delay tolerant network," *Sigcomm Comput. Commun. Rev.*, vol. 34, no. 4, 2004.

[15] K. Lee *et al.*, "Mobile data offloading: How much can WiFi deliver?" ACM Co-NEXT, 2010.

[16] E. Modiano *et al.*, "Maximizing throughput in wireless networks via gossiping," ACM Sigmetrics, 2006.

[17] A. J. Nicholson and B. D. Noble, "Breadcrumbs: Forecasting mobile connectivity," ACM MobiCom, 2008.

[18] S. Nirjon *et al.*, "MultiNets: Policy oriented real-time switching of wireless interfaces on mobile devices," ACM RTAS, 2012.

[19] E. Nordström *et al.*, "Serval: an end-host stack for service-centric networking," in *Proc. USENIX Networked Systems Design and Implementation (NSDI)*, 2012.

[20] M.-R. Ra *et al.*, "Energy-delay tradeoffs in smartphone applications," ACM MobiSys, 2010.

[21] A. Rahmati and L. Zhong, "Context-based network estimation for energy-efficient ubiquitous wireless connectivity," *IEEE Trans. on Mobile Computing*, vol. 10, no. 1, 2011.

[22] P. Rodriguez *et al.*, "MAR: a commuter router infrastructure for the mobile internet," ACM MobiSys, 2004.

[23] T. Spyropoulos *et al.*, "Spray and wait: An efficient routing scheme for intermittently connected mobile networks," in *Proc. ACM SIGCOMM Workshop on Delay-Tolerant Networking,(WDTN)*, 2005.

[24] D. Stiliadis and A. Varma, "Latency-rate servers: a general model for analysis of traffic scheduling algorithms," *IEEE/ACM Trans. Netw.*, vol. 6, no. 5, 1998.

[25] Average Web Page Size Triples Since 2003. [Online]. Available: http://www.websiteoptimization.com/speed/tweak/average-web-page/,2011

[26] M. Woh *et al.*, "AnySP: anytime anywhere anyway signal processing," ACM ISCA, 2009.

[27] P. Yang *et al.*, "Seamless integration of 3G and 802.11 wireless network," ACM MobiWac, 2007.

[28] T. F. Znati and R. Melhem, "Node delay assignment strategies to support end-to-end delay requirements in heterogeneous networks," *IEEE/ACM Trans. Netw.*, vol. 12, no. 5, 2004.