FastHLA: Energy-Efficient Mobile Data Transfer Optimization Based on Historical Log Analysis

Kemal Guner[1] , MD S Q Zulkar Nine[1] , Tevfik Kosar[1] , M. Fatih Bulut[2]

[1] University at Buffalo (SUNY)
(kemalgne, mdsqzulk, tkosar)@buffalo.edu

[2] IBM Thomas J. Watson Research Center
mfbulut@us.ibm.com

ABSTRACT

Mobile data traffic will exceed PC Internet traffic by 2020. As the number of smartphone users and the amount of data transferred per smartphone grow exponentially, limited battery power is becoming an increasingly critical problem for mobile devices which depend on the network I/O. Despite the growing body of research in power management techniques for the mobile devices at the hardware layer as well as the lower layers of the networking stack, there has been little work focusing on saving energy at the application layer for the mobile systems during network I/O. In this paper, we propose a novel technique, called FastHLA, that can achieve significant energy savings at the application layer during mobile network I/O without sacrificing the performance. FastHLA is based on historical log analysis and real-time dynamic tuning of mobile data transfers to achieve the optimization goal. FastHLA can increase the data transfer throughout by up to 10X and decrease the energy consumption by up to 5X compared to state-of-the-art HTTP/2.0 transfers.

1 INTRODUCTION

It is estimated that mobile data traffic will exceed PC Internet traffic the first time in the history, reaching 370 Exabytes per year, by 2020 [47]. A regular smartphone consumes between 300 – 1200 milliwatts power [11] depending on the type of applications it is running, and most of the energy in smartphone applications is spent for networked I/O. During an active data transfer, the cellular and WiFi components of a smartphone consume more power than its CPU, RAM, and even LCD+graphics card at the highest brightness level [11, 40]. Although the mobile data traffic and the amount of energy spent for it increase at a very fast pace, the battery capacities of smartphones do not increase at the same level to meet the demand.

Rapid battery drain is one of the most critical problems for smartphones and mobile computing, and many techniques have been proposed in the literature to overcome this at different layers. At the physical layer, techniques were proposed to choose appropriate modulation, coding, and transmission power control schemes to improve energy efficiency of the mobile device [15, 18, 43, 45]. At the media access control (MAC) layer, several new energy-efficient MAC protocol designs were proposed [10, 32, 52, 55]. At the network layer, low-power and scalable routing algorithms were developed [14, 44, 48, 54]. At the transport layer, traffic shaping techniques [4] and new transport protocols [4, 13, 33, 59] were proposed to exploit application-specific information and reduce power utilization.

Although there has been a growing body of research in power management techniques for the lower layers of the mobile networking stack, there has been little work focusing on saving network I/O (data transfer) energy at the application layer. The most notable work in this area are: tuning the client playback buffer size during media streaming in order to minimize the total energy spent [8]; modeling the parallel communication problem with two proposed energy optimization problems [16]; using lossless compression techniques to minimize the amount of data transferred as well as the energy consumed on wireless devices [53]; and joint optimization of the application layer, data link layer, and physical layer of the protocol stack using an application-oriented objective function in order to improve multimedia quality and power consumption at the same time [28]. We claim that significant amount of network I/O energy savings can be obtained at the application layer with no or minimal performance penalty. Although lower-layer network stack approaches are an important part of the solution, application-layer power management is another key to optimize network

MobiWac'18, Montreal, QC, Canada

DOI: 10.1145/3265863.3265871

I/O energy efficiency in mobile computing, as a complementary approach to the optimizations at the lower layers of the networking stack.

In this paper, we propose a novel technique, called FastHLA, that can achieve significant energy savings at the application layer during mobile network I/O without sacrificing the performance. FastHLA is based on historical log analysis and real-time dynamic tuning of mobile data transfers to achieve the optimization goal. FastHLA can increase the data transfer throughput by up to 10X and decrease the energy consumption by up to 5X compared to state-of-the-art HTTP/2.0 transfers. The improvement is even larger compared to baseline HTTP/1.1 transfers.

The rest of this paper is organized as follows: Section II presents background information on energy-aware tuning of application-layer data transfer protocol parameters and discusses the related work in this area; Section III provides the methodology of our analysis; Section IV introduces our novel FastHLA model and compares it to the competing approaches; and Section V concludes the paper.

2 BACKGROUND

The work on power-aware networking focuses on saving energy at the networking devices. Gupta et al. [24] were among the earliest researchers to advocate conserving energy in networks. They suggested different techniques such as putting idle sub-components (i.e. line cards, etc.) to sleep [23], which were later extended by other researchers. S. Nedevshi et al. [37] proposed adapting the rate at which switches forward packets depending on the traffic load. Other related research in power-aware networking has focused on architectures with programmable switches [22] and switching layers that can incorporate different policies [26]. Barford et al. proposed power-aware network protocols for energy-efficiency in network design and routing [12].

Most of the work on mobile device energy savings focus on putting the devices to sleep during idle times [32, 42, 49, 50]. A recent study by Dogar et al. [21] takes this approach to another step, and puts the device into sleep even during data transfer by exploiting the high-bandwidth wireless interface. They combine small gaps between packets into meaningful sleep intervals, thereby allowing the NIC as well as the device to doze off. Another track of study in this area focuses on switching among multiple radio interfaces in an attempt to reduce the overall power consumption of the mobile device [7, 17, 38]. These techniques are orthogonal to our application-layer protocol tuning approach and could be used together to achieve higher energy efficiency in the mobile systems. The closest work to ours in the literature is the work by Bertozzi et al. [9], in which they investigate the energy trade-off in mobile networking as a function of the TCP receive buffer size and show that the TCP buffering mechanisms can be exploited to significantly increase energy efficiency of the transport layer with minimum performance overheads.

In this work, we focus on the tuning of two application-layer protocol parameters: (1) ***concurrency***, which refers to sending multiple files simultaneously through the network using different data channels at the same time [30, 31, 35]; and (2) ***parallelism***, which sends different chunks of the same file using different data channels (i.e., TCP streams) at the same time and achieves high throughput by mimicking the behavior of individual streams and getting a higher share of the available bandwidth [25, 34, 46]. When used wisely, these parameters have a potential to improve the end-to-end data transfer performance at a great extent, but improper use of these parameters can also hurt the performance of the data transfers due to increased load at the end-systems and congested links in the network. For this reason, it is crucial to find the best combination for these parameters with the least intrusion and overhead to the system resource utilization and power consumption.

In this context, several highly-accurate predictive models [29, 57, 58] were developed which would require as few as three sampling points to provide very accurate predictions for the parallel stream number giving the highest transfer throughput for the wired networks. Yildirim et al. analyzed the combined effect of parallelism and concurrency on end-to-end data transfer throughput [56]. Engin et al.[6] and Nine et al.[39] developed cutting-edge algorithms that consider both historical data analysis and dynamic tuning of the protocol parameters. Alan et al. analyzed the effects of parallelism and concurrency on end-to-end data transfer throughput versus total energy consumption in wide-area wired networks in the context of GridFTP data transfers [5].

3 METHODOLOGY

For the power measurements, we used a single-phase portable Yokogawa WT210 power meter [20], which provides highly accurate and fine granular power values (can measure DC and AC signals from 0.5 Hz to 100 kHz with an accuracy of 99.8%) Prior to initiating any data transfer, we examined the base power state of each tested mobile device. To measure the base power state, we established a setting when the mobile device is in the "on" state with the screen is also on (always at the same brightness level), any communication interface other than the one being tested (i.e., Wifi or 4G LTE) is disabled, and a minimum number of necessary applications is running in the background. This setup ensured that the base power of the tested mobile device is both low and in a balanced state throughput the experiments.

We used four different mobile devices in the experiments (as specifications presented in Table 1). We tested both WiFi and 4G LTE connections in progress of data transfers on

Producer	Google	Samsung	Samsung	Samsung
Model	Nexus S	Galaxy Nexus N3 (L700)	Galaxy S4	Galaxy S5
OS	Android 4.1.1 (API 16)	Android 4.3 (API 18)	Android 5.0.1 (API 21)	Android 5.0.1 (API 21)
CPU	1.0 GHz Cortex-A8	Dual-core 1.2 GHz	Quad-core 1.9 GHz Krait 300	Quad-core 2.5 GHz Krait 400
Wifi	802.11 b/g/n	802.11 a/b/g/n	802.11 a/b/g/n/ac	802.11 a/b/g/n/ac
Storage	16 GB	32 GB	16 GB	16 GB
Memory	512 MB	1 GB	2 GB	2 GB

Table 1: Specifications of the mobile devices used in the experiments.

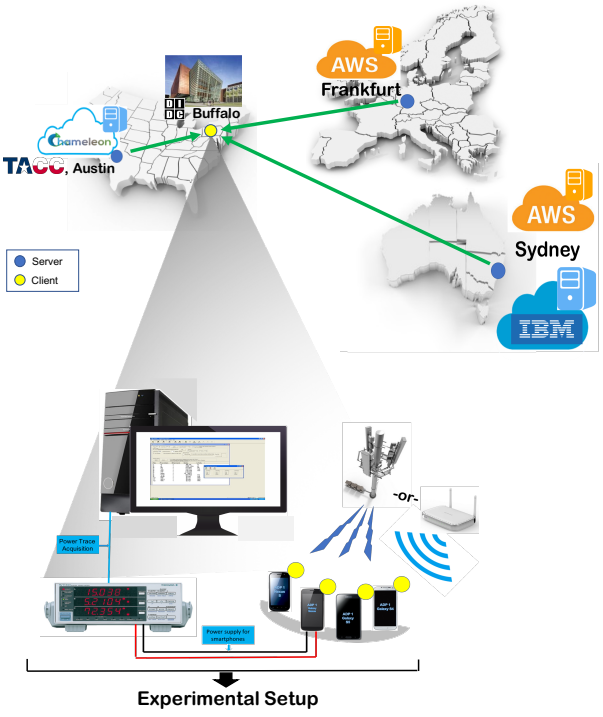


Figure 1: Network map of the experimental testbed and the setup of the power measurement system.

end-systems. To reduce the effect of number of active users and the effect of peak/off-peak hours during the transfer of datasets, we adopted a strategy of using different time frames for each of the same experiment settings, and take the average throughput and energy consumption values. We conducted all experiments at the same location and with the same distance and interference for objective analysis of the end-system devices. We run initial tests for all four mobile devices at different times of the day to obtain robust base power for each. With the help of these values, the total energy consumption during data transfers is calculated as follows:

$$E_t = E_b + E_d \tag{1}$$

Dataset Name	Avg. File Size	Min-Max
HTML	112 KB	56 KB - 155 KB
IMAGE	2.7 MB	2 MB - 3.2 MB
VIDEO-small	152 MB	140 MB - 167 MB
VIDEO-medium	3 GB	2.86 GB - 3.1 GB
VIDEO-large	10GB	9.7 GB - 10.2 GB

Table 2: Characteristics of the dataset used in the experiments for WiFi and 4G/LTE connections.

$$E_d = \int_{t_{start}}^{t_{end}} (P_{max}(t) - P_b(t)) \cdot dt \tag{2}$$

where,

- E_t: Total energy consumption of data transfer
- E_d: Dynamic energy consumption of data transfer
- E_b: Base energy consumption of data transfer
- P_{max}: Total power consumption
- P_b: Base power consumption before initiating the test
- t_{start}: Data transfer start time
- t_{end}: Data transfer end time

Dynamic energy consumption E_d in Equation 2 is established by taking integral of subtract values of base power of device from total instantaneous power measured by power meter per second. All the energy consumption results presented in the paper refer to dynamic energy consumption as stated in Equation 2. Since we aim to analyze the effect of application-layer parameters on energy consumption, we ignored the energy consumed when the device is idle.

We chose HTTP (Hypertext Transport Protocol) as the application-layer transfer protocol to test the impact of the parameters of interest on the end-to-end data transfer throughput as well as the energy consumption of the mobile client. The main reason for this choice is that HTTP is the de-facto transport protocol for Web services ranging from file sharing to media streaming, and the studies analyzing the Internet traffic [19, 27, 41] show that HTTP accounts for 75% of global mobile Internet traffic. We analyzed the data transfer throughput of HTTP data transfers and the power consumption during which we run tests with different levels of concurrency (cc), and parallelism (p). We also measured the instantaneous power consumption and total energy consumption of each individual request among different web

servers and mobile clients. The experiments were conducted on Amazon Elastic Compute Cloud (AWS EC2) [51] instances, Chameleon Cloud [36], and Data Intensive Distributed Computing Laboratory (DIDCLAB). The network map of the experimental testbed and the setup of the power measurement system are illustrated in Figure 1.

In the experiments, we used five different types of files in order to analyze the effect of each individual parameter on transfer throughput and energy consumption. The details and characteristics of these files are presented in Table 2. In order to increase the robustness of the obtained throughput and energy consumption values for each experimental setting, we run each test within the range of five to ten times, and the average values of throughput and energy consumption were used. As a result of iteration of each individual experiment among four different mobile clients and three different web servers with different bandwidth (BW) and round-trip-time (RTT), we transferred varying size of nearly 1.8 Million individual files. Due to the space limitations of the paper, we had to limit the number of graphs we can present. The detailed analysis of the application-layer parameter effects on mobile data transfer performance and energy consumption are provided and discussed in the next section.

4 PROPOSED MODEL

Dynamic nature of the real-time background traffic on network links has a profound impact on the data transfer performance, and makes it very challenging to predict the optimal parameter combination to achieve the highest throughput possible. Historical data transfer logs can provide useful information about the data transfer pattern on a given link and the achievable throughput behavior. We define optimization based on this approach as Historical Log Analysis (HLA). However, the only historical analysis is not enough to keep up with the dynamic network conditions. We also need the current network status to decide and dynamically tune the parameter settings. We define this approach as Online Network Probing (ONP). An ideal solution might be combining both approaches to find the best parameter settings for the transfers. HLA model is an offline analysis model that takes historical transfer logs as input and finds optimal parameter settings for the requested data transfer. However, we have to take into account several design challenges explained below.

Challenge 1. Mobile devices are not suitable for compute-intensive historical analysis. Historical analysis needs to be done outside the mobile device.

Challenge 2. Historical analysis introduces a new cost on both computation and energy consumption. We also have to consider the frequency of the historical analysis as each full iteration of the analysis will introduce more compute and power cost.

Challenge 3. The benefit of figuring out the optimal parameter setting has to outweigh a transfer without any optimization. Assuming C is the cost function, we can strictly constrain C as follows -

$$C(HLA) + C(T_{opt}) < C(T_{no\text{-}opt}) \tag{3}$$

Here, T_{opt} is the transfer with optimized parameters and $T_{no\text{-}opt}$ is the transfer without any optimization.

Challenge 4. Mobile devices have limited memory, therefore, we should allocate a fixed memory size to store the logs. Due to the fixed memory size, it is possible that the new logs can overwrite the old historical logs.

Challenge 5. We also need to make sure that the communication between the historical analysis server and source device is minimal. Too much communication can take toll on the data transfer throughput.

To address these challenges, we have introduced a dynamic framework called *FastHLA* as presented in the next subsection.

4.1 Fast Historical Log Analysis (FastHLA)

Conceptually, FastHLA outsources the analysis of historical data transfer logs to an edge server or to the cloud. The historical analysis will introduce additional computation and energy consumption. Even if we outsource the task it is still consuming computational resources and power on the edge server or in the cloud. *Challenge* 3 might seem counter-intuitive at the beginning, however, we have seen that a transfer with sub-optimal parameter choice achieves low throughput which leads to longer transfer time and high power consumption. We designed FastHLA in a way that it does not need to be run for every transfer. Therefore, the cost of FastHLA can be amortized over many subsequent transfers. To generalize the model even farther we can run FastHLA for many mobile devices in the cloud to amortize the FastHLA cost over many mobile devices.

An overview of the FastHLA model is provided in Figure 2. We introduced a light-weight transfer broker that receives a transfer request and performs transfer with best possible parameters. Network condition does not change significantly over a short period of time, however, when it changes the previous optimal choice of parameters might become sub-optimal. Therefore, running FastHLA once in the beginning is no better than statically setting the transfer parameters (an approach used in many current solutions). Therefore, we need a strategy to minimize the frequency of running FastHLA. To resolve this issue, we introduced a caching mechanism for previous optimal parameters and a Learning

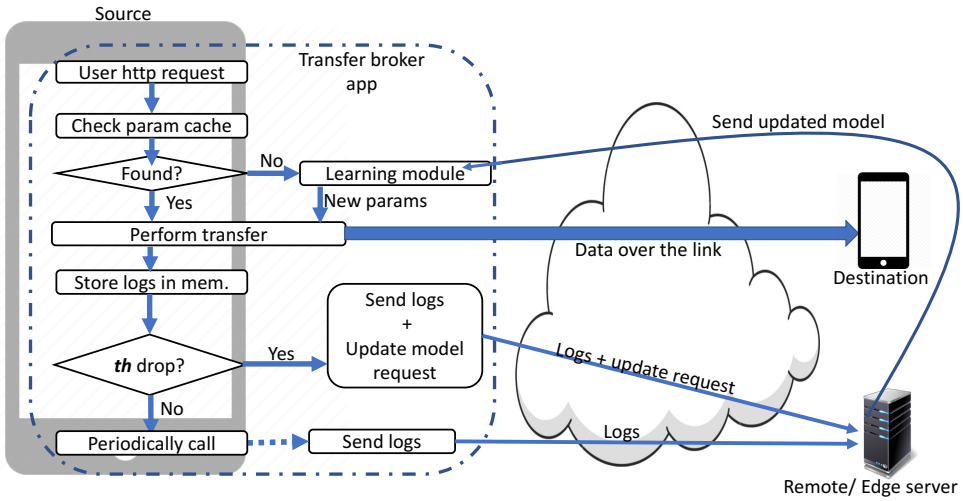


Figure 2: Overview of the FastHLA model.

Module (LM) in the mobile device. The parameter cache is a dictionary structure that maps network condition to parameter value list. On the other hand, the LM can take user request and network condition as input and provide best known parameters. The training of the LM is performed as a part of FastHLA outside the mobile device. Each time the FastHLA runs, it updates the local Learning Module. A trained LM in the mobile device can provide predicted optimal parameters in almost constant time. Transfer broker first looks into the cache for the parameter settings. In case of a cache miss, it asks parameters from the LM and performs the transfer using those parameters. It is crucial to keep LM up-to-date. It can be done by accepting periodic updates from FastHLA. However, there might be a highly unlikely case where the network condition is unknown to the LM itself and the parameters provided by LM are sub-optimal. In that case, an immediate update request will be issued only if there is a significant drop in data transfer performance. However, the chance of such miss significantly reduces after each FastHLA update, because the training is an additive process with proper generalization method. Therefore, LM gets more and more precise after each FastHLA update.

To address memory issue explained in *Challenge* 4, the transfer broker periodically sends historical transfer logs to the remote server where HLA is performed, so that old historical logs become available to HLA before being overwritten by the new logs. The communication overhead explained in *Challenge* 5 are the periodic update requests and periodic log transfers. We do not need to communicate with HLA server during the transfer except during the highly unlikely case explained above where both cache and LM fails to provide optimal parameters. To reduce the communication overhead during the transfer, we decided not to include the Online Network Probing into our model. We can use the most recent logs to assess the network condition.

The model consists of five steps - (1) historical transfer log collection and preprocessing; (2) clustering similar logs; (3) optimization; (4) learning optimal parameters; and (5) scheduling mixed sized data. The details of these steps are explained below.

Step 1 – Historical log collection and preprocessing: We collect historical logs for the previous data transfers. Historical log contains detailed information about the data, network characteristics, application level parameters, mobile device information, and external traffic status. Data information contains file size (fs) and number of files (n_{files}). Network characteristics contain round trip time (t_{rtt}), tcp buffer size (bs_{tcp}), and bandwidth (bw). Application level parameters contain concurrency (cc), parallelism (p) and I/O block size (b_{io}). Resource usage information contains CPU utilization (μ_{cpu}), memory utilization (μ_{mem}), NIC card utilization (μ_{nic}), and power consumption (pw). Mobile device information contains the model, operating system, processor, memory, and network interface (WiFi/radio) specifics.

Historical logs might contain information about the transfers which were aborted or failed; or sometimes, due to a system error, logs might contain unreasonable information such as achieved throughput greater than the bandwidth. During preprocessing phase we remove those logs. *Standard outlier detection* model is used to remove those outliers.

Step 2 – Clustering similar logs: Application level parameters have different impact on different types of transfers. Categorize logs into groups based on their similarity could provide us more structured view of the log information. After analyzing the logs we come to the conclusion that some

parameters have direct precedence over other parameters. We use *Hierarchical Agglomerative Clustering* which is the most suitable clustering technique for such cases.

Step 3 – Optimization: This is the most important part of the analysis. We first modeled both throughput and energy function based on historical log. Then we performed mathematical optimization to find the best parameter settings. The details of the optimization are presented in Section 4.2.

Step 4 – Learning optimal parameters: As we decided to do historical log analysis on the edge servers or in the cloud, there should be an efficient way to transfer the acquired knowledge from the analysis server to the mobile devices. The simple solution is sending the optimal results gained for each transfer to the mobile device. But, this is not a scalable solution as this approach is too specific to the individual transfers. A mobile device cannot generalize the knowledge for even similar transfers. Moreover, it will take a considerable amount of memory to store those individual results. Another solution would be the use of machine learning techniques, which can be used to learn the knowledge from the optimization step, and can have the power to predict parameters for the unknown transfers.

Machine learning techniques come with two distinct steps – (1) learning and (2) prediction. These two steps can be decoupled. As all the historical logs and optimization results are stored in HLA servers, it is reasonable to do learning step in the HLA server. Then the trained model is transferred to the mobile device. Another reason is to choose the number of parameters (also known as weights, connections) in the learning module, which is fixed. The number of connections and weights do not increase as the historical log increases, only the values of the weights are updated as the model learns. Therefore, HLA server always sends a fixed sized update (e.g., updated values of the weights) to the mobile device. It simultaneously optimizes the memory and communication overhead between the server and the device. In our model we have used off the shelf non-linear machine learning techniques, such as - Artificial Neural Networks (ANN) and Support Vector Machines (SVM). As we have limited feature space (number of meta-data in the log), we do not need any deep learning techniques capable of extracting complex pattern from high number of features.

Step 5 – Scheduling mixed sized data: We observed that the files with different sizes can have different optimal parameter settings. Therefore, a dataset containing different sized files should not be transferred with the same parameter settings. A more fine-tuned solution is to cluster the files based on similarity and use optimal parameter settings for each cluster. However, each optimal parameter setting is optimized for that specific cluster and agnostic towards other clusters' parameters. Transferring these clusters concurrently can over-provision the network and introduce packet loss. Therefore, we scale down the parameter values according to the the cluster size and some known heuristics. An overview is provided in Algorithm 1.

Algorithm 1: Mixed Data Scheduling

input : Data arguments, $data\_args = \{Dataset, avg\_file\_size, num\_files\}$
output : Optimal parameter settings, θ_{opt}

procedure Scheduling(F_s, R_s, I_s)
  $C \leftarrow$ cluster($Dataset$)
  for c_i **in** C **do**
    $\theta_i \leftarrow$ get_optimal_params (c_i)
  end
  if sum $(\theta_i) >$ user_limit **then**
    $\theta_{opt} \leftarrow (\theta_i \times \text{user\_limit})/\text{sum}(\theta_i)$
  end

4.2 Optimization

Application level parameters, such as concurrency (cc), parallelism (p) and I/O block size (bs) can be tuned properly to achieve both high throughput and low energy consumptions. We define throughput (th) and energy consumption ($\mathbb{E}$) as:

$$th = f_{th}(p, cc, bs) \tag{4}$$

$$\mathbb{E} = f_e(p, cc) \tag{5}$$

Appropriate modeling of th and $\mathbb{E}$ is crucial to find the optimal parameter settings. Historical log contains samples of the parameter space, therefore, can not provide overall view of the whole parameter space. We need an interpolation technique to predict the missing parameters. Then we can optimize these functions to get optimal parameters. We follow two steps - (1) interpolation of unknown parameters, and (2) finding the optimal parameters. These steps are explained below.

Step 1 – Interpolation of unknown parameters: We observed that the throughput and energy consumption follow a cubic pattern. Therefore, we modeled both throughput and energy consumption as piece-wise cubic interpolation. Cubic interpolation fills the achievable throughput and energy consumption of the unknown parameters. These piece-wise cubic functions are stitched with a guarantee of smoothness up to second derivative. As I/O block size is different from concurrency and parallelism, we modeled it separately.

To model the throughput, we construct a 2-dimension cubic spline interpolation for $th = f(bs)$. Piece-wise cubic interpolation can be constructed using interpolant $th_i = f(bs_i)$ and connecting them by maintaining smoothness up

to second derivative. We can define each cubic polynomial piece as generic cubic function:

$$f_i(bs) = x_{i,0} + x_{i,1}bs + x_{i,2}bs^2 + x_{i,3}bs^3, \forall bs \in [bs_i, bs_{i+1}]. \quad (6)$$

Boundaries can be constrained as $f(bs_{i+1}) = f(bs_i)$. We can have:

$$f_i(bs_i) = th_i, \quad i = 1, ..., N \quad (7)$$

Therefore, the N continuity constraints of $f(bs)$ are as:

$$f_{i-1}(bs_i) = th_i = f_i(bs_i), \quad i = 2, ..., N. \quad (8)$$

The following constraint confirms smoothness up to second derivatives.

$$\frac{d^2 f_{i-1}}{d^2 bs}(bs_i) = \frac{d^2 f_i}{d^2 bs}(bs_i), \quad i = 2, ..., N \quad (9)$$

The boundary condition for spline could be written as:

$$\frac{d^2 f}{d^2 bs}(bs_1) = \frac{d^2 f}{d^2 bs}(bs_n) = 0 \quad (10)$$

The coefficients can be computed by solving the system of linear equations.

Throughput is also dependent on concurrency and parallelism. The example above can be extended to generate throughput surface with two independent variables - *cc* and *p*. Similarly, we modeled energy as a function of concurrency, parallelism and I/O block size.

Step 2 – Find optimal parameters: Energy efficient transfer aims to reduce the energy consumption without compromising the transfer performance. This objective function tries to optimize both throughput and power consumption at the same time. This objective function does not guarantee both maximally achievable throughput with minimum power consumption, however, it ensures that every unit of power can be spent to achieve highest possible throughput under the energy efficiency constraint. The objective function here is to maximize achievable throughput over power consumption.

$$\text{maximize} \quad \int_{t_s}^{t_f} th/\mathbb{E} \quad (11)$$

We take into account all the boundary constraints for the parameters and other necessary constraints. Due to the space limitation we are not including those here. Then we used non-linear optimizer to find the optimal parameters.

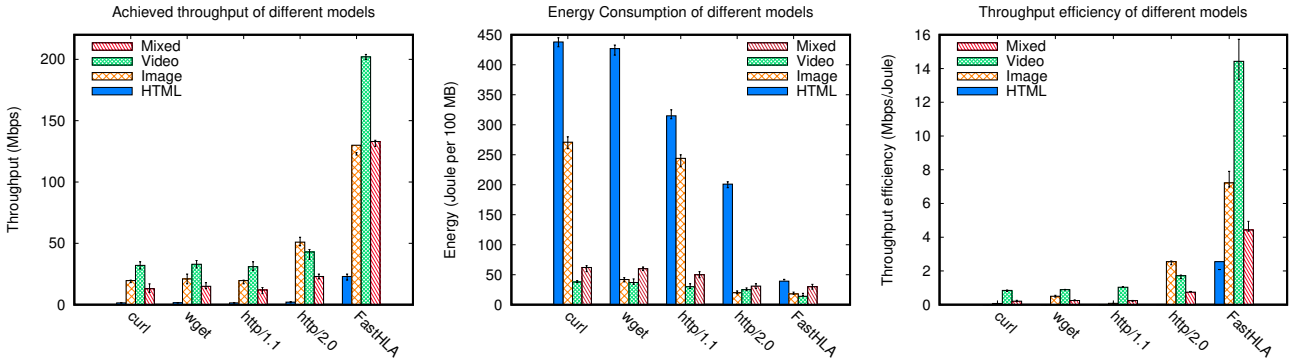
4.3 Evaluation of FastHLA Model

Historical log analysis (HLA) data transfer experiments are conducted in the same experimental testbed described in Section 3. We trained and tested our model (FastHLA) on the real data transfer logs and compared the performance and power consumption FastHLA with energy-agnostic wget [3] and curl [1] clients as well as two versions of de-facto application layer transfer protocol of HTTP, which are HTTP/1.1 and HTTP/2 [2]. While HTTP/1.1 is a textual protocol, the newly introduced HTTP/2 is a binary protocol that supports multiplexing, header compressions and lets the server to *push* responses.

To evaluate our model, we used HTML, image and video datasets (as described in Section 3) along with a combined dataset that contains a mix of three datasets. We compared FastHLA with other models using these datasets so that we can get a fine-grained analysis of performance. Figure 3 shows both energy and throughput comparison of different existing approaches along with our model. We observe that FastHLA outperforms all other tested solutions in every data category. For image, video, HTML, and mixed data sets, we see 2×, 4×, 10× and 4× throughput improvement over the closest competitor HTTP/2. HTTP/2 uses multiplexing to transfer multiple streams over a single connection to remove head-of-line blocking. However, single connection can achieve very poor results in long RTT WAN links. HTTP/1.1 uses multiple connections to request multiple files, however, there is no way to dynamically set those number of connections (parallelism). On the other hand, we have used historical analysis to decide an optimal level of parallelism, concurrency and I/O block size.

Figure 3(b) shows the energy consumption of different models. As we can see standard applications like wget or curl are not optimized for power consumption and draw a huge energy compare to HTTP/2. On the other hand, FastHLA improves power consumption 5× and 2× for HTML and video files respectively compared to HTTP/2. We have observed that high transfer throughput can shorten the data transfer time. That means CPU has to work for a shorter period of time and CPU consumes most of the power during the transfer. That explains why FastHLA consumes less power compared to other approaches. However, the energy consumption is similar for image and mixed data. Even if the power consumption is similar for image and video, FastHLA can provide more achievable throughput compared to HTTP/2. We use throughput efficiency, $th/\mathbb{E}$ to measure the energy efficiency of the models (as shown in Figure 3(c)). FastHLA improves throughput efficiency 2.5× for both video and image data.

We have used three different learning modules to see the efficiency of those models. Among them ANN and SVM can reach up to 94% and 92% accuracy respectively (as shown in Figure 4). However, KNN can achieve up to 86% accuracy. As K-Nearest Neighbor takes into account k closest logs to decide on the parameters, it is not feasible to transfer all the optimal results to mobile due to memory issues. That is why we decided not to use KNN in our model. However, the Neural Network and SVM both can learn efficiently the



(a) Achieved Throughput **(b) Energy Consumption** **(c) Throughput Efficiency**

Figure 3: Achievable throughput and corresponding energy consumption of different optimization objectives and the accuracy of the learning module.

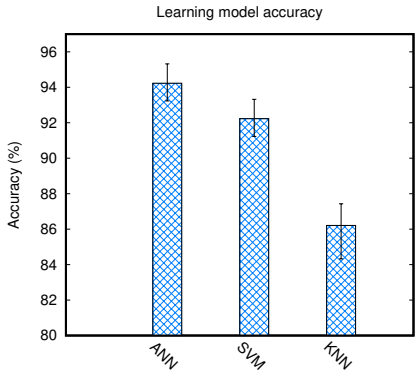


Figure 4: Accuracy of different learning modules.

optimal parameters. We can use coefficient of determination, R^2 to see how well the prediction of learning module matches the target. It can be expressed as:

$$R^2 = 1 - \sum(y_i - \bar{y})^2 / \sum(f_i - \bar{y})^2 \quad (12)$$

where y_i is the actual optimal throughput and $\bar{y}$ is the mean of y_i. The predicted throughput is defined as f_i. R^2 is a good statistical indicator that can point out the strongly actual and prediction values are related. In case of perfect matches between all known targets and the predictions R^2 value will be 1. However, we can say a model can predict with good generalization if R^2 value is close to 1. We computed this for both ANN and SVM as $R^2_{ANN} = 0.92$ and $R^2_{SVM} = 0.87$ respectively.

5 CONCLUSION & FUTURE WORK

In this paper, we proposed a novel historical-data analysis based model, called FastHLA, that can achieve significant energy savings at the application layer during mobile network I/O without sacrificing the performance. Our analysis shows that FastHLA model can achieve significant energy savings using only application-layer solutions at the mobile systems during data transfer with no performance penalty. We also show that, in many cases, our FastHLA model can increase the performance and save energy simultaneously.

According to our experiments, by intelligently tuning the concurrency and parallelism levels during data transfers, our FastHLA model can increase the data transfer throughput by up to 10X, and decrease the energy consumption by up to 5X compared to state-of-the-art HTTP/2.0 transfers. The improvement is even larger compared to base HTTP/1.1 transfers and client tools such as wget and curl.

As a future work, we are planning to develop service-level-agreement (SLA) based transfer tuning algorithms to balance the performance vs energy trade-off during mobile network I/O according to the preferences of the mobile users. While keeping the quality of service (i.e., transfer throughput) at the desired level, these algorithms will try to keep the energy consumption at the minimum possible level.

ACKNOWLEDGEMENTS

This project is in part sponsored by the National Science Foundation (NSF) under award numbers OAC-1724898 and OAC-1842054, and by IBM Research under award number OCR-W1771224. We also would like to thank Chameleon Cloud and AWS for letting us use their resources for some of the experiments presented in this paper.

REFERENCES

[1] curl. http://curl.haxx.se/.
[2] Okhttp http/2 client for android. http://square.github.io/okhttp/.
[3] wget. https://www.gnu.org/software/wget/.
[4] S. A. Akella, R. K. Balan, and N. Bansal. Protocols for low-power. 2001.
[5] I. Alan, E. Arslan, and T. Kosar. Power-aware data scheduling algorithms. In *Proceedings of IEEE/ACM (SC15)*, November 2015.
[6] E. Arslan, K. Guner, and T. Kosar. Harp: Predictive transfer optimization based on historical analysis and real-time probing. In *Proceedings of IEEE/ACM conference SC'16*, pages 288–299.

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

[8] D. Bertozzi, L. Benini, and B. Ricco. Power aware network interface management for streaming multimedia. In *IEEE Wireless Communications and Networking Conference, WCNC2002*.

[9] D. Bertozzi, A. Raghunathan, L. Benini, and S. Ravi. Transport protocol optimization for energy efficient wireless embedded systems. In *Proceedings of the conference on Design, Automation and Test in Europe-Volume 1*, page 10706. IEEE Computer Society, 2003.

[10] V. Bharghavan, A. Demers, S. Shenker, and L. Zhang. Macaw: a media access protocol for wireless lan's. *ACM SIGCOMM Computer Communication Review*, 24(4):212–225, 1994.

[11] A. Carroll and G. Heiser. An analysis of power consumption in a smartphone. In *USENIX annual technical conference*, 2010.

[12] J. Chabarek, J. Sommers, P. Barford, C. Estan, D. Tsiang, and S. Wright. Power awareness in network design and routing. In *In Proceedings of IEEE INFOCOM, April, 2008*.

[13] S. Chandra and A. Vahdat. Application-specific network management for energy-aware streaming of popular multimedia formats. In *USENIX Annual Technical Conference, General Track*, pages 329–342, 2002.

[14] J.-H. Chang and L. Tassiulas. Energy conserving routing in wireless ad-hoc networks. In *INFOCOM 2000*, volume 1, pages 22–31, 2000.

[15] E. Cianca, M. Ruggieri, and R. Prasad. Improving tcp/ip performance over cdma wireless links: A physical layer approach. In *Personal, Indoor and Mobile Radio Communications, 2001.*

[16] M. Conti, B. Crispo, D. Diodati, J. K. Nurminen, C. M. Pinotti, and T. Teemaa. Leveraging parallel communications for minimizing energy consumption on smartphones. *IEEE Transactions on Parallel and Distributed Systems*, 26(10):2778–2790, Oct 2015.

[17] L. M. Correia, D. Zeller, O. Blume, D. Ferling, Y. Jading, I. Gódor, G. Auer, and L. Van Der Perre. Challenges and enabling technologies for energy aware mobile radio networks. *Communications Magazine, IEEE*, 48(11):66–72, 2010.

[18] S. Cui, A. J. Goldsmith, and A. Bahai. Energy-efficiency of mimo and cooperative mimo techniques in sensor networks. *IEEE Journal on selected areas in communications*, 22(6):1089–1098, 2004.

[19] J. Czyz, M. Allman, J. Zhang, S. IekelJohnson, E. Osterweil, and M. Bailey. Measuring ipv6 adoption. *SIGCOMM Comput. Commun. Rev.*, 44(4):87–98, Aug. 2014.

[20] K. M. Dixit. Overview of the spec benchmarks., 1993.

[21] F. R. Dogar and P. Steenkiste. Catnap: Exploiting high bandwidth wireless interfaces to save energy for mobile devices. In *Proc. Int. Conf. Mobile Systems, Applications and Services (MobiSys)*, 2010.

[22] A. Greenberg, P. Lahiri, D. A. Maltz, P. Patel, and S. Sengupta. Towards a next generation data center architecture: Scalability and commoditization. In *In ACM PRESTO, pages 57ÃŘ62, 2008*.

[23] M. Gupta and S. Singh. Energy conservation with low power modes in ethernet lan environments. In *IEEE INFOCOM (MiniSymposium) 2007*.

[24] M. Gupta and S. Singh. Greening of the internet. In *ACM SIGCOMM, pages 19ÃŘ26, 2003*.

[25] T. J. Hacker, B. D. Noble, and B. D. Atley. Adaptive data block scheduling for parallel streams. In *Proceedings of HPDC '05*, pages 265–275. ACM/IEEE, July 2005.

[26] D. A. Joseph, A. Tavakoli, and I. Stoica. A policy-aware switching layer for data centers. In *SIGCOMM CCR 38(4):51ÃŘ62, 2008*.

[27] A. Kellerman. *Daily spatial mobilities: Physical and virtual*. Routledge, 2016.

[28] S. Khan, Y. Peng, E. Steinbach, M. Sgroi, and W. Kellerer. Application-driven cross-layer optimization for video streaming over wireless networks. *IEEE Communications Magazine*, 44(1):122–130, 2006.

[29] J. Kim, E. Yildirim, and T. Kosar. A highly-accurate and low-overhead prediction model for transfer throughput optimization. In *Proc. of DISCS Workshop*, November 2012.

[30] T. Kosar and M. Balman. A new paradigm: Data-aware scheduling in grid computing. *Future Generation Computing Systems*, 25(4):406–413.

[31] T. Kosar and M. Livny. Stork: Making data placement a first class citizen in the grid. In *Proceedings of ICDCS'04*, pages 342–349.

[32] R. Krashinsky and H. Balakrishnan. Minimizing energy for wireless web access with bounded slowdown. *Wireless Networks*.

[33] R. Kravets and P. Krishnan. Application-driven power management for mobile communication. *Wireless Networks*, 6(4):263–277, 2000.

[34] J. Lee, D. Gunter, B. Tierney, B. Allcock, J. Bester, J. Bresnahan, and S. Tuecke. Applied techniques for high bandwidth data transfers across wide area networks. In *International Conference on Computing in High Energy and Nuclear Physics*, April 2001.

[35] W. Liu, B. Tieman, R. Kettimuthu, and I. Foster. A data transfer framework for large-scale science experiments. In *Proc. 3rd International Workshop on Data Intensive Distributed Computing (DIDC '10) in conjunction with (HPDC '10)*, June 2010.

[36] J. Mambretti, J. Chen, and F. Yeh. Next generation clouds, the chameleon cloud testbed, and software defined networking (sdn). In *Cloud Computing Research and Innovation (ICCCRI), 2015 International Conference on*, pages 73–79. IEEE, 2015.

[37] S. Nedevschi, L. Popa, G. Iannaccone, S. Ratnasamy, and D. Wetherall. Reducing network energy consumption via rate-adaptation and sleeping. In *Proceedings Of NSDI, April 2008*.

[38] A. Nika, Y. Zhu, N. Ding, A. Jindal, Y. C. Hu, X. Zhou, B. Y. Zhao, and H. Zheng. Energy and performance of smartphone radio bundling in outdoor environments. In *Proceedings of the 24th International Conference on World Wide Web*, pages 809–819, 2015.

[39] M. S. Q. Z. Nine, K. Guner, Z. Huang, X. Wang, J. Xu, and T. Kosar. Big data transfer optimization based on offline knowledge discovery and adaptive sampling. In *Big Data 2017*, pages 465–472.

[40] A. Pathak, Y. C. Hu, and M. Zhang. Where is the energy spent inside my app?: fine grained energy accounting on smartphones with eprof. In *Proceedings of the 7th ACM european conference on Computer Systems*, pages 29–42. ACM, 2012.

[41] P. Richter, N. Chatzis, G. Smaragdakis, A. Feldmann, and W. Willinger. Distilling the internet's application mix from packet-sampled traffic. In *Passive and Active Measurement*, pages 179–192.

[42] 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 sixteenth annual international conference on Mobile computing and networking*, pages 85–96. ACM, 2010.

[43] C. Schurgers, O. Aberthorne, and M. Srivastava. Modulation scaling for energy aware communication systems. In *Proceedings of the 2001 international symposium on Low power electronics and design*, pages 96–99. ACM, 2001.

[44] K. Seada, M. Zuniga, A. Helmy, and B. Krishnamachari. Energy-efficient forwarding strategies for geographic routing in lossy wireless sensor networks. In *Proceedings of the 2nd international conference on Embedded networked sensor systems*, pages 108–121. ACM, 2004.

[45] S. Singh and C. S. Raghavendra. PamasÃŚpower aware multi-access protocol with signalling for ad hoc networks. *ACM SIGCOMM Computer Communication Review*, 28(3):5–26, 1998.

[46] H. Sivakumar, S. Bailey, and R. L. Grossman. Psockets: The case for application-level network striping fpr data intensive applications using high speed wide area networks. In *SC'2000*.

[47] C. Systems. Visual networking index: Forecast and methodology, 2015–2020, June 2016.

[48] C.-K. Toh. Maximum battery life routing to support ubiquitous mobile computing in wireless ad hoc networks. *IEEE communications Magazine*, 39(6):138–147, 2001.

[49] N. Vallina-Rodriguez and J. Crowcroft. Erdos: achieving energy savings in mobile os. In *Proceedings of the sixth international workshop on MobiArch*, pages 37–42. ACM, 2011.

[50] N. Vallina-Rodriguez and J. Crowcroft. Energy management techniques in modern mobile handsets. *Communications Surveys & Tutorials, IEEE*, 15(1):179–198, 2013.

[51] J. Varia and S. Mathew. Overview of amazon web services. *Amazon Web Services*, 2014.

[52] A. Woo and D. E. Culler. A transmission control scheme for media access in sensor networks. In *Proceedings of the 7th annual international conference on Mobile computing and networking*, pages 221–235, 2001.

[53] R. Xu, Z. Li, C. Wang, and P. Ni. Impact of data compression on energy consumption of wireless-networked handheld devices. In *Distributed Computing Systems, 2003. Proceedings. 23rd International Conference on*, pages 302–311. IEEE, 2003.

[54] Y. Xu, J. Heidemann, and D. Estrin. Geography-informed energy conservation for ad hoc routing. In *Proceedings of the 7th annual international conference on Mobile computing and networking*, pages 70–84. ACM, 2001.

[55] W. Ye, J. Heidemann, and D. Estrin. An energy-efficient mac protocol for wireless sensor networks. In *INFOCOM 2002. Twenty-First Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings. IEEE*, volume 3, pages 1567–1576. IEEE, 2002.

[56] E. Yildirim, E. Arslan, J. Kim, and T. Kosar. Application-level optimization of big data transfers through pipelining, parallelism and concurrency. *To appear in IEEE Transactions on Cloud Computing (TCC)*, 18(1):41–59, 2015.

[57] E. Yildirim, D. Yin, and T. Kosar. Prediction of optimal parallelism level in wide area data transfers. *IEEE Transactions on Parallel and Distributed Systems*, 22(12), 2011.

[58] D. Yin, E. Yildirim, and T. Kosar. A data throughput prediction and optimization service for widely distributed many-task computing. *IEEE Transactions on Parallel and Distributed Systems*, 22(6), 2011.

[59] M. Zorzi and R. R. Rao. Is tcp energy efficient? In *Mobile Multimedia Communications, 1999.(MoMuC'99) 1999 IEEE International Workshop on*, pages 198–201. IEEE, 1999.