NORTHWESTERN
UNIVERSITY

Electrical Engineering and Computer Science Department

Technical Report
NWU-EECS-10-06
March 26, 2010

Characterizing and Modeling User Activity on Smartphones

Alex Shye, Benjamin Scholbrock, Gokhan Memik, and Peter A. Dinda

Abstract

We present a comprehensive analysis of real smartphone usage during a 6-month study, involving 25 users, and one specific smartphone, the Android G1. The goal of our work is to study the high-level characteristics of smartphone usage, and to understand the implications on optimizing smartphones, and their networks. We present 11 findings that cover general usage behavior, interaction with the battery, power consumption, network activity, frequently-run applications, and modeling usage states. We then discuss the implications of these findings on optimizing and/or modeling smartphones and their networks. Some of our findings include the following: (1) we show that battery management is a significant part of the user experience, with near daily charging and frequent low battery indicators; (2) we characterize the system-level power consumption to show that the screen and CPU are two of the most power hungry components on a smartphone; (3) we characterize EDGE and wifi session, providing insight into developing models for smartphone networks; and (4) we demonstrate that real user activity can be clustered to yield a small number of machine states, and state transitions, that can form the basis for modeling smartphone devices.

Keywords: Smartphones, Human Factors, Measurements, Embedded Devices

Characterizing and Modeling User Activity on Smartphones

Alex Shye, Benjamin Scholbrock, Gokhan Memik, Peter A. Dinda
Department of Electrical Engineering and Computer Science
Northwestern University
Evanston, IL
{shye, scholbrock, memik, pdinda}@northwestern.edu

ABSTRACT

We present a comprehensive analysis of real smartphone usage during a 6-month study, involving 25 users, and one specific smartphone, the Android G1. The goal of our work is to study the high-level characteristics of smartphone usage, and to understand the implications on optimizing smartphones, and their networks. We present 11 findings that cover general usage behavior, interaction with the battery, power consumption, network activity, frequently-run applications, and modeling usage states. We then discuss the implications of these findings on optimizing and/or modeling smartphones and their networks. Some of our findings include the following: (1) we show that battery management is a significant part of the user experience, with near daily charging and frequent low battery indicators; (2) we characterize the system-level power consumption to show that the screen and CPU are two of the most power hungry components on a smartphone; (3) we characterize EDGE and wifi session, providing insight into developing models for smartphone networks; and (4) we demonstrate that real user activity can be clustered to yield a small number of machine states, and state transitions, that can form the basis for modeling smartphone devices.

1. INTRODUCTION

Advances in computing and communication technology have recently converged in the form of *mobile smartphones*. Mobile smartphones, or smartphones for short, are mobile phones integrated with PC-like hardware and software, i.e., a microprocessor, storage devices, wifi, and a full-blown operating system. The unique combination of portable computation and communication not only enables general-purpose applications on mobile phones, but also enables a new class of mobile applications such as push email and location-aware social network services. Similar to mobile phones, which were adopted faster than any technology in history [1], the market for smartphones is expected to grow significantly in coming years, with predictions of outselling PCs by as early as 2011 [17].

Looking forward, there are two trends we can expect: (1) a significant growth in the smartphone market, and (2) a constant demand from the market for more functionality. Increasing functionality requires computation and communication, both of which consume energy. However, the form factor of a smartphone limits battery size, and, ultimately, the available energy store. Smartphones must be optimized for energy efficiency, in order to provide more functionality while maintaining a reasonable battery life. In addition, the rapid growth in the number of devices, and an increase in the traffic per device, will drastically increase network traffic load. This effect has already been demonstrated, with the load of iPhone users on AT&T's wireless data network [11][19]. Thus, it will also be increasingly important to optimize the operation of the communication networks supporting smartphones.

The first step towards optimization is usually to identify and characterize representative workloads for the target computing platform. Workload characterization is critical for identifying components that should be targeted for optimization, as well as finding trends or patterns in the workload to develop optimizations.

In the case of a smartphone, what is the workload? Although it is possible to answer this question at several levels, our answer is simple: the workload of a smartphone is the end user. This is intuitive; the activity of a smartphone is either driven directly by user interaction, or indirectly via the user's network or environment. For a more concrete example, let us consider two different users; the first uses the device primarily as a web browser, and the second uses the device solely as a phone. If our goal is to reduce power consumption, we would focus on optimizing the general-purpose microprocessor for the first user, but ignore it for the second user (smartphones typically have a dedicated modem processor for the cellular phone functionality). While this example may be simplistic, it demonstrates the importance of considering the usage of the end user when characterizing the system-level workload of a smartphone.

Despite the importance of the end user in understanding workloads, there does not exist a comprehensive study for analyzing and modeling user behavior at a system-level on smartphones. To fill this need, we collect real smartphone usage from real users on real devices, and present a comprehensive analysis of the data from several perspectives. In all, we collect logs of of real smartphone usage from a 6-month study, involving 25 users, and one specific smartphone, the Android G1. The main goal of our study is to observe the high-level workload characteristics of real smartphone users in the wild, and understand the implications of these characteristics for optimizing smartphones, and their networks.

.

General Observations (Section 3)	**Implications**
1. Users recharge their phones on a daily basis, and use their phones until the battery is low in $\sim$ 20% of the cases when it is unplugged for over 4 hours.	Battery management is a significant part of the smartphone user experience, including daily charging and frequent low battery indicators.
Improving Energy Efficiency (Section 4)	**Implications**
2. There is significant variation in the usage behavior of our users. For example, User 4 is a heavy phone user, User 17 mostly leaves the phone in *Idle* mode, and User 22 is a heavy wifi user.	It is critical to perform real user studies when studying smartphones. There are large variations across our users that are manifested as significant changes in power consumption.
3. The *Active* state consumes 53.7% of the total system power, even though it only accounts for 11% of the usage time. Of the hardware components, the screen and the CPU consume the most power, 19.5% and 7.3% of the total power, respectively.	The *Active* consumes the majority of the power, even though it accounts for a small fraction of the total use time. The screen and CPU require the most attention with respect to energy efficiency.
4. On average, the phone is in the *Idle* state 89% of the time and accounts for 46.3% of the total system power.	Reducing the power consumption of the *Idle* state should also be a high priority for improving battery life.
5. Most users do not switch between multiple brightness levels, nor do they install power management software.	Automatic brightness adjusting optimizations should be included with smartphones. Considering the magnitude of system power attributable to the screen among all hardware components, there is an underutilized opportunity for power optimization.
6. The CPU utilization is typically either at 100%, or under 10%.	Dynamic CPU scale-down optimizations, such as dynamic voltage and frequency scaling, should be used for saving power.
Networking (Section 5)	**Implications**
7. EDGE network session durations follow a power-law distribution, and the network traffic is highly dependent on the time of day.	Session durations can be modeled with a General Pareto Distribution, with a a shape parameter of 2.7454 and a scale parameter of 2.4732.
8. Wifi network session durations appear to be the sum of several distributions and the wifi network traffic is highly dependent on daily usage modes.	Modeling wifi session durations warrants more investigation and shows promise of revealing trends in user behavior. Session durations can be modeled with a MMPP.
Application Usage (Section 6)	**Implications**
9. A significant portion of CPU utilization is attributable to OS-level processes.	From the perspective of mobile computing, OS developers must be aware of the broader impacts of frequently-run code (e.g., on power consumption due to the CPU).
Usage Patterns (Section 7)	**Implications**
10. From a large space of possible states, only a few significant states and transitions are required to meaningfully represent smartphone usage patterns.	Building a useful state-transition graph to model smartphone user behavior from a large dataset is tractable.
11. Automatic clustering of usage logs closely matches manually selected states of interest.	Meaningful states may be extracted automatically from input data in order to build a state-based model of user behavior.

Table 1: Summary of our findings, including references to the section containing the details of each finding.

Specifically, we are interested in the following questions:

- What does typical user activity look like? How does it differ across users?
- What are the most power consuming hardware components? What are the execution characteristics of the most power hungry components?
- What are the network connectivity characteristics on a smartphone?
- How can we detect high-level patterns in user activity?

We first discuss our methodology, including a logger application we have developed for studying user activity and our user selection process (Section 2). We then present general characteristics of the data that is logged, as well as battery usage (Section 3). Afterwards, we characterize the total system power consumption, identify the hardware components that consume the most power, and provide an in-depth analysis of execution characteristics for these hardware components (Section 4). We explore the characteristics of EDGE and wifi traffic on mobile phones and their implications with respect to modeling smartphones (Section 5). We present the commonly used application in Section 6. Finally, we demonstrate that commonly-seen phone usage states can be automatically extracted, via clustering, and can be used to build a model of user transitions through several usage patterns (Section 7). We present related work in Section 8, and conclude the paper in Section 9.

For the convenience of the reader, we summarize our findings in Table 1. The table includes our 11 main observations, their implications energy-efficient design and optimization, and forward references to the sections describing each observation.

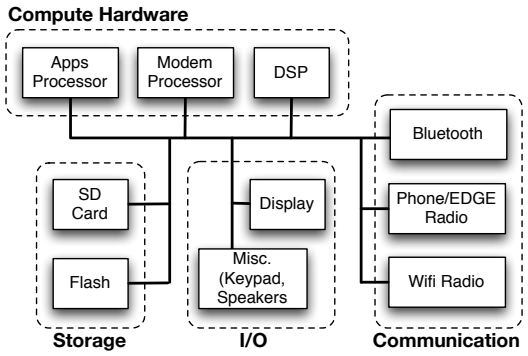

Figure 1: High-level overview of the target mobile architecture.

2. METHODOLOGY

In this section, we provide a general overview of the Android G1, our logger application, and the power model we use for estimating power consumption of the Android G1.

2.1 The Android G1

Our target smartphone in this paper is the HTC Dream, marketed as the Android G1, a cellular phone platform built by HTC that supports the open source Google Android mobile device platform [6]. At the time of our studies, there were two Android G1 phones; the G1 Android Developer Phone (ADP1), and the commercially released T-Mobile G1 phone. Both phones are identical, except that the ADP1 is a rooted and SIM-unlocked version for developers. Although we focus on a specific smartphone, our contributions and findings could easily extend to other devices.

The software on the Android G1 is the open-source Android platform, which consists of a slightly modified 2.6 Linux kernel, and a general framework of C, C++, and Java code. The framework includes the Dalvik Virtual Machine (VM), a variant of Java implemented by Google. All userspace applications are Dalvik executables which run in an instance of the Dalvik VM.

A high-level diagram of the Android G1 is shown in Figure 1. The ADP1 has a 3.2inch HVGA 65K color capacitive touch screen, uses a Qualcomm MSM7201A chipset, and a 1150mAh lithium-ion battery [18]. The Qualcomm MSM7201A chipset contains a 528MHz ARM 11 apps processor, a ARM 9 modem processor, QDSP4000 and QDSP5000 high-performance digital signal processors, 528MHz ARM 11 Jazelle Java hardware acceleration, quadband GPRS and EDGE network, integrated bluetooth, and wifi support.

To the best of our understanding, the ARM 11 apps processor runs the Android platform and executes the applications on the device. It is rated at 528MHz and supports dynamic frequency scaling (DFS), but is scaled down in the platform to run at 124MHz, 246MHz, and 384MHz. The highest frequency of 528MHz is not used. The ARM 9 modem processor is a separate processor that runs a proprietary operating system and is in charge of the communications of the phone. The Jazelle Java hardware acceleration processor is not used as the Android platform runs Dalvik executables which are not fully compatible.

2.2 Logging User Activity

To study the real usage of the Android G1, we have developed a logger application that logs user activity events, as well as system-level performance measurements. The logger is developed as a normal Dalvik executable using the Java standard libraries available in the Android framework. Thus, it runs on both of the Android G1 devices without any special hardware or OS support. At a high-level the logger application consists of two parts; (1) a GUI application which looks like a normal Android GUI application (complete with a clickable icon), and (2) an associated background service to provide logging functionality. We release the logger on Android Market, the portal for Android users to browse and download Android applications.

The GUI application. When opened by the user, the GUI application begins the background service and displays information about the logger on the screen. The information includes an overview of our project, a complete list of the information we capture, a disclaimer regarding user consent for volunteering their log data, a start/stop button to control the background service, and a link to the project webpage including project information and F.A.Q.

Background service. The background service implements the logging functionality, continually logging user behavior and system performance characteristics. It periodically looks for a network connection and sends the anonymized logs logs back to our server.

A complete list of the data in the traces is shown below, including the method of data collection, and the sampling interval for sampled data. Note that the sampling interval accounts for active phone time. Thus, if the phone is in an idle mode, we do not sample anything. If not listed, the data is collected via a Android platform library call or callback. The data is categorized by represented hardware component, if applicable, from Figure 1.

- Apps Processor
  - CPU utilization from `/proc/stat` [1sec]
  - System load from `/proc/loadavg` [1sec]
- DSP
  - Whether media (music / video) is playing
- Display
  - Display brightness and state (on/off)
- Phone/EDGE Radio
  - Phone state (airplane mode, signal strength)
  - Phone call state (ringing, call, idle)
  - Data plan connectivity (i.e., EDGE)
  - Bytes sent/received from EDGE network from `/proc/self/net/dev` [1sec]
- Wifi Radio
  - Wifi state (connectivity, signal strength)
  - Bytes sent/received from wifi network from `/proc/self/net/dev` [1sec]
- SD Card
  - SD Card traffic from `/proc/self/net/dev` [1sec]
- Other
  - Opening and closing of the logger application
  - Location, from the cellular network and GPS (only sampled when the user triggers the Android location services) [30sec]
  - Battery charging state
  - Statistical sample of application usage similar to `top` Unix/Linux utility (via `/proc/pid/stat`) [5min]

2.3 User Selection

To obtain users for the study, we posted online advertisements and physical flyers for anonymous volunteers on various university campuses, technical news web sites, and Android-related forums. The only prerequisite for the study is that users own an Android G1 phone. Overall, we collect logs from 52 users from April–November 2009. For this paper, we use the logs from the 25 users with the longest total recorded time. The data from these 25 users represents approximately 1329 days ($\sim$ 3.6 years) of real user activity, with an average of approximately $\sim$ 53 days of logged user activity per user.

Any user-based study should be carefully designed to reduce sources of bias, including observer bias and selection bias. Observer bias refers to the bias induced by the observer. Selection bias refers the bias induced when using a non-random volunteer selection process.

On observer bias. For our study, there are two potential sources of observer bias. First, it is possible that the users may change their behavior because they know that they are being logged. To minimize this source of bias, we were completely transparent with our volunteers. The GUI application was clear about the data collected. The Android Market and logger application both contained links to an email address, and a web page for the project. We responded to all questions received via email and maintained a frequently asked questions section on the webpage. We also guaranteed anonymity. We made it clear that all data was anonymized and that it was impossible for us to trace any data to a specific person. The second potential source of bias comes from the logger application. If it significantly perturbs the usage of the phone, it may change user behavior. In order to minimize impact on user experiences, we implemented the logger application to incur a low overhead. The logger typically incurs less than 4% CPU overhead during active phone use, and only consumes network bandwidth to transfer log data. Before transferring log data, we compressed the logs with `gzip` to minimize network traffic.

On selection bias. With respect to selection bias, we did not attempt to control the selection or demographics of our participants. The users are anonymous volunteers responding to advertisements posted on multiple university campuses and various publicly viewable online communities. We must note that the anonymous nature of this process has pros and cons with respect to selection bias. On one hand, it removes our potential biases in selecting participants. On the other hand, our process also makes it impossible for us to guarantee or track any information about the volunteer pool. When considering these tradeoffs, we believed that the benefits of anonymous user selection outweighed the drawbacks, especially considering that our anonymous approach also aids in minimizing the observer bias.

2.4 Power Estimation

We use a regression-based power estimation model for the Android G1 that has been proposed and validated in prior literature [16]. We first transform our log data into a form amenable to the power model. This involves processing the logs, which contain different types of samples (asynchronous events and timer-based samples), to create the samples to feed the power model. Each sample is a vector that contains the various parameters needed in the power model. We then use the power estimation model to accurately estimate the total system power consumption, as well as derive a power breakdown between the hardware components.

User	# Sessions	Total Time	Hrs/Session
User1	7	454.6 hrs	64.9 hrs
User2	11	529.3 hrs	48.1 hrs
User3	131	1060.8 hrs	8.1 hrs
User4	12	297.2 hrs	24.8 hrs
User5	145	2784.2 hrs	19.2 hrs
User6	242	572.2 hrs	2.4 hrs
User7	7	361.0 hrs	51.6 hrs
User8	34	3415.6 hrs	100.5 hrs
User9	11	435.2 hrs	39.6 hrs
User10	58	422.3 hrs	7.3 hrs
User11	94	1770.5 hrs	18.8 hrs
User12	47	657.8 hrs	14.0 hrs
User13	84	725.1 hrs	8.6 hrs
User14	190	3178.2 hrs	16.7 hrs
User15	101	1093.9 hrs	10.8 hrs
User16	362	2479.1 hrs	6.8 hrs
User17	84	3368.8 hrs	40.1 hrs
User18	96	909.1 hrs	9.5 hrs
User19	41	1132.1 hrs	27.6 hrs
User20	9	449.0 hrs	49.9 hrs
User21	131	3216.0 hrs	24.5 hrs
User22	11	257.8 hrs	23.4 hrs
User23	126	939.4 hrs	7.5 hrs
User24	35	1032.1 hrs	29.5 hrs
User25	6	354.9 hrs	59.1 hrs
Total	2075	1329.0 days	28.5 hrs

Table 2: Statistics on logger sessions per user.

At a high level, the power model splits the phone usage into two modes, *Active* and *Idle*. During *Active* time, the apps processor and/or the DSP processor is in use. To estimate the power consumption during *Active* time, the power model uses parameters that correspond to characteristics of different hardware components on the smartphone, including he CPU, screen, cellular phone radio, EDGE/wifi radios, etc. It includes a catch-all `system` parameter for the components that the model does not take into account (y-intercept for the linear regression). During *Idle* time, the power consumption is small, and relatively constant. The model uses a constant of $\sim$ 70 mW for each of these samples. We process the traces from the logger to divide each trace into samples for the power estimation model. We use a 4 second sliding window to generate a sample of the system-level parameters for every second. We have performed experiments in the lab for various usage scenarios and found that the absolute error rate for these scenarios were, on average, 6.6%.

3. GENERAL USAGE BEHAVIOR

In this section, we provide an overall picture of the time intervals within our log data, and then present general behavior on battery usage.

3.1 Logger Sessions

We first discuss the characteristics of the time intervals that we are able to account for. The logger does not always provide a steady stream of data. There may be gaps in time that we are unable to account for, such as when the phone is shut down, the operating system fails, the battery is removed, etc. Thus, we first process our logs is to extract

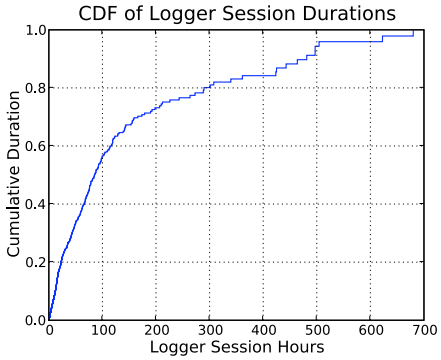

Figure 2: CDF of logger session intervals.

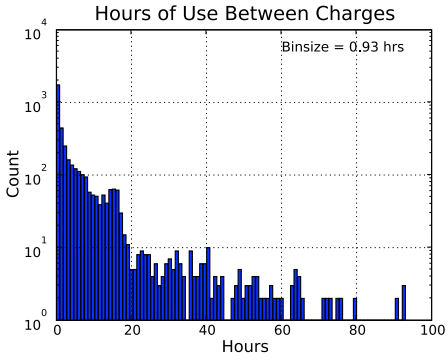

Figure 3: Histogram of battery intervals when the phone is not charging.

Intervals of n+ hrs	% Time Unplugged	# Times Unplugged	# Battery Low (%)
0+ hrs	67.3%	3852	417 (10.8%)
2+ hrs	68.2%	1625	273 (16.8%)
4+ hrs	68.2%	1213	236 (19.5%)

Table 3: Summary statistics on battery usage.

logger sessions, time intervals we know we can account for based upon the log data.

Table 2 shows statistics on the logger sessions for each of our users. The table includes the number of logger sessions, the total time of the logger sessions, and the average logger session time. Overall, we extract 2075 logger sessions that account for approximately 1329 days of user activity. On average, our logger sessions account for 28.5 hours of continuous data samples. There is a high variation in average logger session time, ranging from 2.4 hours for User 6 to 100.5 hours for User 8. Although we cannot be certain, we believe that the users with very short logger session are most likely Android developers who tend to restart their phones often (as we know from experience in developing the logger). Although there are many short logger sessions, they do not account for a considerable fraction of the total logged time. Figure 2 shows a CDF representing the logger session time versus the total cumulative time. The CDF shows that over 80% of the logged time can be attributed to logger sessions that account for over 24 consecutive hours. The short logger sessions that are under 1 hour account for 43% of the logger sessions, but less than 1% of the logged time.

3.2 Battery Usage

To study the battery usage of our users, we extract *battery intervals* from each of the logger sessions. A battery interval represents a time interval where we can account for the battery being plugged or unplugged. Figure 3 shows a histogram showing the distribution of unplugged battery intervals based upon their duration (note the y-axis has log scale). Although there are some very long intervals, the majority of unplugged battery intervals are less than 20 hours. This demonstrates that most of our users recharge their phones at least on a daily basis.

Table 3 shows battery interval statistics, with each row representing the set of battery intervals with a time duration of n+ hours. For each row, we show the percent of time unplugged, the number of unplugged battery intervals, and the number of unplugged battery intervals that included a low battery indicator (the phone notifies the user when less than 15% of the battery remains). Overall, the phone is unplugged $\sim$ 68% of the time. Many of the unplugged battery intervals are filtered out when we remove the short intervals (over half of the intervals when filtering out intervals less than 2 hours). In addition, the low battery indicator appears a significant number of times – once for every 3.2 days of recorded phone use – in spite of the trend of daily charging.

Finding 1: Users recharge their phones on a daily basis, and use their phones until the battery is low in $\sim$ 20% of the cases when it is unplugged for over 4 hours.
Implication: Battery management is a significant part of the smartphone user experience, including daily charging and frequent low battery indicators.

4. CHARACTERIZING G1 USAGE: POWER PERSPECTIVE

We now characterize the Android G1 usage from a power perspective. In this section, we are primarily interested in the unplugged battery intervals (as described in Section 3.2) since they represent regular battery-constrained usage. We first present a time breakdown of usage, and then present a power breakdown to characterize the most power consuming hardware components. We then study the two most power hungry components in more detail: the screen, and the CPU.

4.1 Time Breakdown

To account for the time in the unplugged battery intervals, we divide each interval into *Idle* time and *Active* time. The *Active* time is further divided between time when the screen is on, and when the screen is off (if there is music playing, the phone may be in *Active* mode with the screen off). The results are shown in Figure 5. Not surprisingly, the phone is *Idle* most of the time (89%), and in the *Active* mode for significantly less (11%). During *Active* time, the screen is on 8% of the time, and is off the 3% of the time.

4.2 System-Level Power Breakdown

Next, we study the total system power breakdown using the power estimation model discussed in Section 2.4. The power breakdown is useful in providing a high-level view of user activity with respect to power consumption, and can be used to determine the hardware components that consume the most power.

The power breakdown from each of the users is shown in Figure 4. The x-axis represents each of the users and

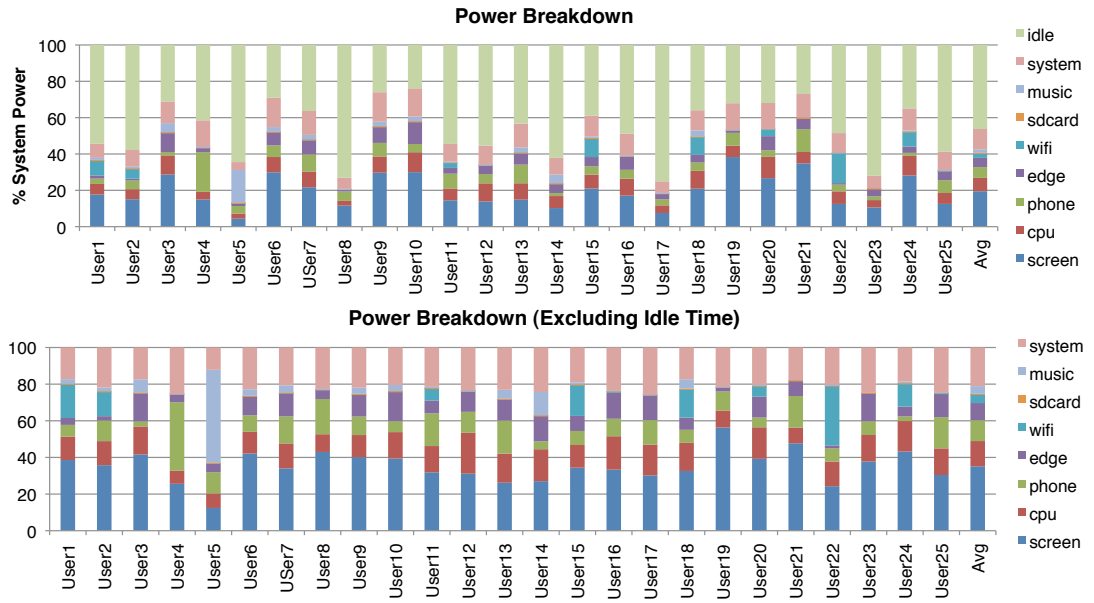

Figure 4: Total system power breakdown with and without considering the *Idle* state time.

the y-axis represents the percentage of total system power. The product terms for each of the hardware components are combined for the power breakdown. To provide a clear breakdown in the *Active* state, Figure 4 also shows the power breakdown omitting the *Idle* state.

Observing the power breakdown in combination with the time breakdown (described in Section 4.1) across our users results in three findings.

Finding 2: There is significant variation in the usage behavior of our users. For example, User 4 is a heavy phone user, User 17 mostly leaves the phone in *Idle* mode, and User 22 is a heavy wifi user.
Implication: It is critical to perform real user studies when studying smartphones. There are large variations across our users that are manifested as significant changes in power consumption.

Finding 3: The *Active* state consumes 53.7% of the total system power, even though it only accounts for 11% of the usage time. Of the hardware components, the screen and the CPU consume the most power, 19.5% and 7.3% of the total power, respectively.
Implication: The *Active* consumes the majority of the power, even though it accounts for a small fraction of the total use time. The screen and CPU require the most attention with respect to energy efficiency.

Finding 4: On average, the phone is in the *Idle* state 89% of the time and accounts for 46.3% of the total system power.
Implication: Reducing the power consumption of the *Idle* state should also be a high priority for improving battery life.

Although the *Idle* state may often dominate the total system power, in this paper, there are three reason to be concerned with the *Active* state. First, the power consumed during the *Idle* state ($\approx$ 68 mW) is orders of magnitude lower than the power that can be consumed in the *Active* state (up to 2000+ mW). Second, the *Active* state contributes highly to the user experience since the user is actively engaged during the *Active* state. Any application that requires the apps processor would require the device to wake up and exit Idle mode. Finally, the *Active* state still accounts for the majority of the total system power.

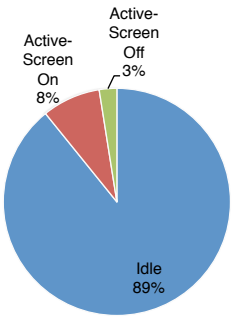

Figure 5: Time breakdown when the phone is not charging.

4.2.1 Screen

As discussed in the previous section, the screen consumes a significant percentage of the total system power during *Active* time. The screen power can be divided into two major components: the backlight power and the power due to the screen brightness. Although the screen backlight power is constant, the screen brightness consumes about $\sim$ 50% of the screen power and can be adjusted to trade-off brightness with battery life. Table 4 shows the number of changes in screen brightness, the number of discrete brightness levels (0 (low)–255 (high)), and the average brightness used by each user over time. Of the 25 users, 16 use one screen brightness, 6 manually switch between 2–7 brightness levels, and 3 use automatic screen adjustment (they have significantly more brightness changes than the other users, signifying an automatic mechanism).

User	# Brightness Changes	# Brightness Levels	Average Brightness
User1	0	1	150.0
User2	0	1	102.0
User3	0	1	133.0
User4	0	1	102.0
User5	11	7	84.3
User6	0	1	102.0
User7	0	1	102.0
User8	1	2	143.4
User9	0	1	102.0
User10	0	1	102.0
User11	2	2	100.8
User12	67	5	52.3
User13	0	1	75.0
User14	2	3	67.1
User15	0	1	102.0
User16	2	3	65.1
User17	0	1	46.0
User18	202	7	105.7
User19	0	1	192.0
User20	0	1	102.0
User21	0	1	255.0
User22	0	1	30.0
User23	162	4	61.2
User24	18	12	141.8
User25	0	1	57.0

Table 4: Summary statistics on screen brightness. Brightness levels on the Android G1 range from 0 (lowest) to 255 (highest).

Finding 5: Most users do not switch between multiple brightness levels, nor do they install power management software.
Implication: Automatic brightness adjusting optimizations should be included with smartphones. Considering the magnitude of system power attributable to the screen among all hardware components, there is an underutilized opportunity for power optimization.

4.2.2 CPU

We next study the CPU activity, as it is the second largest power consumer during the *Active* state. Figure 6 shows a histogram of the CPU utilization samples across all of our users during the unplugged battery intervals when the screen is on. Note that our logger application causes a very small overhead, which shifts the CPU utilization up by 1 to 3%. For the most part, the CPU utilization is significantly divided between high and low utilization. The phone is typically at 100% utilization or below 10% utilization, spending the majority of time in the latter case. As discussed in Section 2.1, the CPU frequency control on the Android G1 is naive, only controlled by the screen state. These results indicate that a more aggressive frequency scaling scheme may save significant power.

Finding 6: The CPU utilization is typically either at 100%, or under 10%.
Implication: Dynamic CPU scale-down optimizations, such as dynamic voltage and frequency scaling, should be used for saving power.

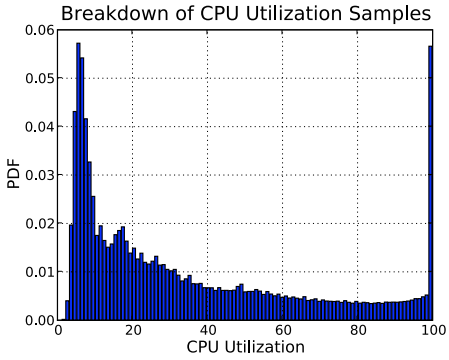

Figure 6: Histogram of CPU utilization samples across all users.

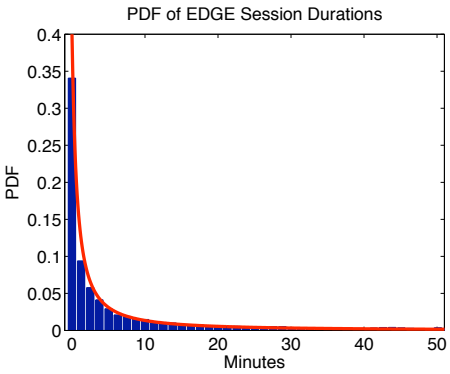

Figure 7: Probability distribution function of EDGE network session durations. The tail of plot is cut off for readability. The red line corresponds to our fit to a General Pareto Distribution, described in Section 5.1.

5. CHARACTERIZING THE G1: NETWORK PERSPECTIVE

Understanding network behavior is important for modeling the networks that support smartphones. In this section, we study EDGE and wifi characteristics for the entire logger sessions, including both plugged and unplugged time. We do this because all of the device traffic is of interest to network designers.

5.1 EDGE Network

We extract 9044 EDGE sessions from all of our users, totaling 1142 days worth of connectivity and averaging 181.9 minutes per session. The EDGE network shared resources with the cellular networking, and thus, is connected the majority of the time, covering 86% of all logged time. Overall, the EDGE network traffic is highly dominated by the inbound traffic, which accounts for 82.3% of the bytes transferred to/from the device.

5.1.1 EDGE Session Durations

Figure 7 shows the probability distribution function (PDF) for the session durations across all of our users. The data follows a power-law distribution, with many short durations that dominate the samples. We also note that the data has a long tail, with single samples in the range of hundreds and even thousands of minutes. We omit them from the graph to improve readability. The PDF matches very closely to

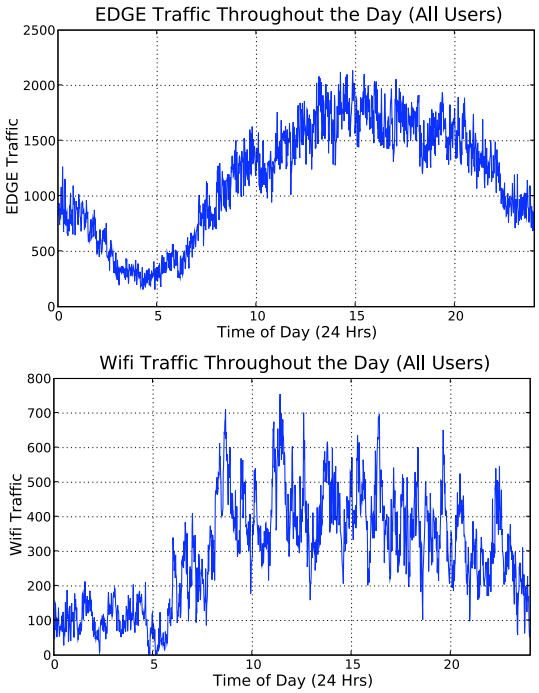

Figure 8: EDGE and wifi network traffic with respect to the time of day (x-axis on a 24-hour timescale).

a General Pareto Distribution with a shape parameter of 2.7454 and a scale parameter of 2.4732. We have verified this fitting with a correlation of determination, R^2, of 0.96.

5.1.2 *EDGE Traffic*

Figure 8 shows the EDGE network traffic accumulate over all the users, when first mapped to the users' local time of day. The network traffic follows a daily pattern, with light traffic during sleep hours (between midnight and 7am for this user) and heavy, bursty, traffic during the day time, including peak activity patterns between 7–10am and 1–3pm. Similar patterns are shared across all users, indicating that there are predictable daily modes of EDGE traffic usage. These traits indicate that a Markov-Modulated Poisson Process (MMPP), which covers each of the main phases of usage behavior and makes decisions based on time of day, may be effective for modeling EDGE network traffic activity.

Finding 7: EDGE network session durations follow a power-law distribution, and the network traffic is highly dependent on the time of day.
Implication: Session durations can be modeled with a General Pareto Distribution, with a a shape parameter of 2.7454 and a scale parameter of 2.4732.

5.2 Wifi Network

We extract 6173 wifi connectivity sessions, totaling 37.6 days worth of connectivity and averaging 8.8 minutes per session. Since wifi is dependent upon an open wireless access point, as well as wifi enabled (which may require explicit user control), the wifi connectivity is not as pervasive as the EDGE network connectivity, and the session durations are significantly shorter. Of our users, only 9 had significant

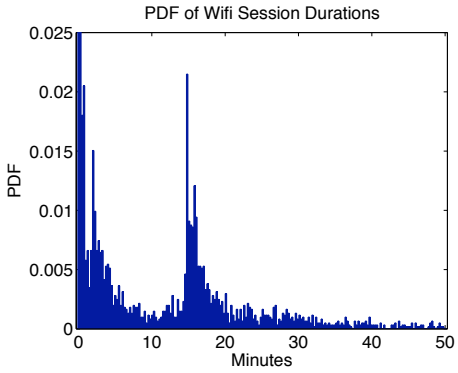

Figure 9: Probability distribution function of wifi network session durations.

Type Program	# Wifi Sessions	Mean/Max Time (min)	Mean/Peak KB/s
Light	2608	13.8/865	3.6/514
Medium	2226	4.3/1003	11.6/1027
Heavy	1017	8.6/604	57.9/2506

Table 5: Types of wifi sessions based upon average traffic: light (< 10 KB/s), medium (< 65 KB/s), and heavy (> 65 KB/s).

patterns of wifi usage. The wifi network traffic is highly dominated by the inbound traffic, which accounts for 88.3% of the bytes transferred to/from the device.

5.2.1 *Wifi Session Durations*

We now study the durations of wifi sessions. Figure 9 shows the PDF of wifi session durations across our users. We should note that the spike in the left-most data point is cut off and goes up to $\sim 50\%$. Also, we omit a long tail, with sparse single samples out to hundreds and thousands of minutes. As shown in the figure, the PDF for wifi is considerably more complex than the EDGE sessions. At a high level, it appears to be a combination of a power law distribution and a bimodal Poisson-like distribution with peaks around 2.5 and 15 minutes. We believe that the two peaks may be due to the aggregate of two main types of wifi sessions: short sessions and long sessions. Short sessions may be due to activities such as checking the weather, while long sessions may include web surfing.

5.2.2 *Wifi Traffic*

Figure 8 shows the wifi network traffic aggregated across all users and mapped to the users' local time of day. The wifi traffic also exhibits daily time-based patterns that almost line up directly with the EDGE network traffic. However, compared to the EDGE daytime traffic, the wifi daytime traffic appears more noisy. Similar patterns are shared across all 9 wifi users, indicating that a MMPP may also be effective for modeling wifi traffic.

We can use wifi traffic to differentiate between different types wifi sessions. Figure 10 shows a histogram of the wifi session binned by their average network traffic. There are two significant inflection points in the graph (marked by the dotted black lines), near 10 and 65 KB/s, that divide the sessions into light, medium, and heavy traffic sessions. We provide more details on these three categories in Table 5, showing the total number of wifi sessions, the mean/max time duration, and the mean/peak throughput in each of

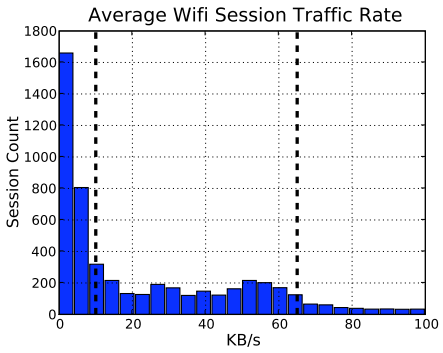

Figure 10: Histogram of the average traffic rate for wifi sessions. The dotted lines specify the breakdown between three categories of sessions: light traffic (< 10 KB/s), medium traffic (< 65 KB/s), and heavy traffic (> 65 KB/s).

these categories. From these results, we see that there are two times more light/medium sessions than heavy sessions, and that mean and peak traffic both clearly distinguish between these categories. We also learn that the medium category, with a short mean duration and a long maximum duration, makes it difficult to use duration as a distinguishing factor between wifi sessions.

Finding 8: Wifi network session durations appear to be the sum of several distributions and the wifi network traffic is highly dependent on daily usage modes.
Implication: Modeling wifi session durations warrants more investigation and shows promise of revealing trends in user behavior. Session durations can be modeled with a MMPP.

6. APPLICATION USAGE

Android devices support a variety of programs downloadable through the Android Market. The Android Market groups programs into *Applications* and *Games*. To get some idea about how often these programs are actually used on the devices, the logger periodically records information similar to the common `top` command (henceforth referred to as `top` samples for simplicity). Considering only the `top` samples when the screen is on, these log entries represent a random sampling of how often programs are used, and how much available computation power they consume.

These two metrics are represented as the percent of `top` samples containing the program, and the percent of CPU utilization reported by `top`. Programs are categorized as:

- **Applications:** Non-Game Programs
- **Games:** Game Programs
- **Phone/MMS:** Phone and MMS processes
- **OS:** OS-level processes

Programs not available in the Android Market which are not games, such as T-Mobile Self Help, provided on T-Mobile phones, are grouped into *Applications*. Only samples which occur when the screen is on and the phone is not charging are considered.

The types of programs observed by `top`, with corresponding average CPU utilization and frequency observed, is shown in Table 6. The logging program is isolated from other *Applications*, and not represented in these numbers.

Program Type	Seen in % of `top` Samples	Average CPU Utilization (%)
Applications	15.89%	2.63%
Games	0.10%	27.77%
Phone/MMS	91.22%	1.14%
OS	100%	11.38%

Table 6: Types of programs in `top` samples.

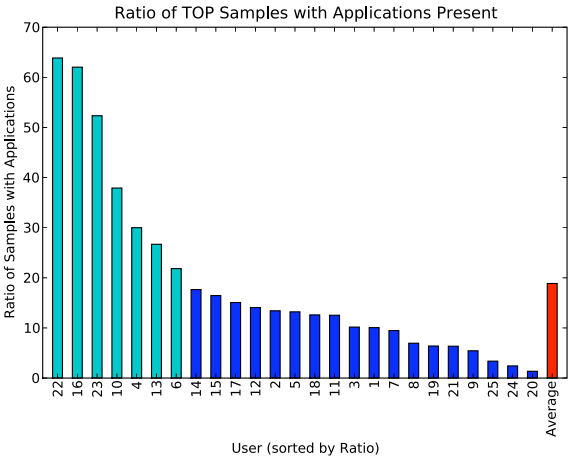

Figure 11: Ratio of `top` Samples with *Applications*, Per-User.

OS processes must be present in each sample; all Android user-level programs run as virtual machines, and the virtual machine server process is part of the OS. Therefore, whenever a program runs (including the logging program and UI programs), the server process must also be running. Likewise, to receive events regarding new messages and calls, the Phone and MMS programs keep daemon processes running with occasional small CPU utilization. Thus, these process types are present in most, if not all, of the samples (shown in the middle column of Table 6).

Figure 11 shows the ratio of `top` samples, for each individual user, which include an Application as defined above. In the figure, the red bar represents the ratio averaged across users, light-blue bars represent users whose ratio is *above* the average, and blue bars represent users whose ratio is *below* the average.

The data shows that, on average, significant contributions to the CPU load by *Applications* occur in only 16% of the samples. Furthermore, when *Applications are* present in a sample, the average combined CPU utilization by all Application processes is 2.6%. *Games* contribute an even smaller portion of the workload, being present in a tiny fraction of the samples (0.1%).

The overall effect on CPU utilization by *Applications* and *Games* are actually significantly smaller than the effect of the OS-level processes. These processes are present in *all* samples, with average CPU utilization of 11.4%.

To some degree, this observation may be skewed by external factors. For example, while the Android Marketplace does have a section for games, and the platform supports OpenGL for game development, the selection of games pales in comparison to other mobile gaming platforms. If people

Cluster	Weights	Call Off-Hook	Call Ringing	Wifi On	Cell Data Active	Wifi Data Active	Screen On	SD Card Active	Media Playing	CPU Util. *Medium Frequency*	CPU Util. *High Frequency*	Corresponding Reduced Cluster
												Manually Selected
1	14.9%	1										1 – Phone Call
2	0.6%	0	1									2 – Phone Ringing
3	12.7%	0	0	1		1						3 – Wifi w/Traffic
4	3.8%	0	0	1		0						4 – Wifi w/out Traffic
5	15.9%	0	0	0	1							5 – Cell w/Traffic
6	4.7%	0	0	0	0				1			6 – Media On
7	47.4%	0	0	0	0				0			7 – Misc.
												Automatically Selected
0	7%				1		1				65.21	5 – Cell w/Traffic
1	3%						1		1		19.75	6 – Media On
2	3%	1								8.24		1 – Phone Call
3	3%	1					1			5.86	25.51	1 – Phone Call
4	7%						1			18.09	6.17	7 – Misc. (∼ 6% CPU)
5	3%								1	15.41	4.17	6 – Media On
6	4%			1			1			4.68	4.64	4 – Wifi w/out Traffic
7	12%						1			1.52	8.91	7 – Misc. (∼ 9% CPU)
8	1%	1		0.06	1	1					52.57	
9	5%						1			2.16	66.67	7 – Misc. (∼ 67% CPU)
10	14%			1		1	1				10.12	3 – Wifi w/Traffic
11	3%						1			25.97	12.98	7 – Misc. (∼ 13% CPU)
12	1%						0.2	1		14.68	37.43	
13	0%						0.11			45.6	4.28	
14	2%				1		1		1	9.57	51.81	5 – Cell w/Traffic
15	5%						1			2.73	20.9	7 – Misc. (∼ 21% CPU)
16	19%						1			5.56	0.94	7 – Misc. (∼ 1% CPU)
17	9%				1		1			8.03	41.74	5 – Cell w/Traffic
18	1%			1		1	1				60.9	3 – Wifi w/Traffic

Table 7: Manually and automatically selected states.

become more likely to play games on their Android device than as observed in this study, CPU utilized by gameplay would correspondingly be much more significant.

Still, with the OS-level processes consuming one tenth of the available computing capabilities, this represents a large portion of the workload.

Finding 9: A significant portion of CPU utilization is attributable to OS-level processes.
Implication: From the perspective of mobile computing, OS developers must be aware of the broader impacts of frequently-run code (e.g., on power consumption due to the CPU).

7. DISCOVERING AND MODELING USAGE PATTERNS

In order to better understand how people use mobile platforms, a useful representation for the data would be *usage states*, specifying how the device is being used at any recorded time. With such a representation, state changes and common patterns over time may be observed more clearly. To develop such a model, we analyze the *Active* state, since this represents time the user is interacting with the phone.

For representing usage states, a subset of potentially meaningful fields are selected:

- Phone on call (off-hook)
- Phone ringing
- Wifi on
- Wifi active (communicating)
- Cell data active (communicating)
- Screen on
- SD card being read/written
- Media (music / video) being played
- CPU utilization (at medium CPU frequency and high CPU frequency)

7.1 Selecting Representative States

To map samples to states, a set of states is chosen as a representative set of meaningful states. This essentially maps a vector of the above values (as boolean) to activities, such as a phone call, or listening to music. The mapping is performed as a binary tree, splitting on one variable at a time. A list of these states, and the values they represent, is shown in Table 7.

With a mapping from gathered statistics to meaningful usage states, a time-series representation of the statistics may be converted to a series of transitioning usage states.

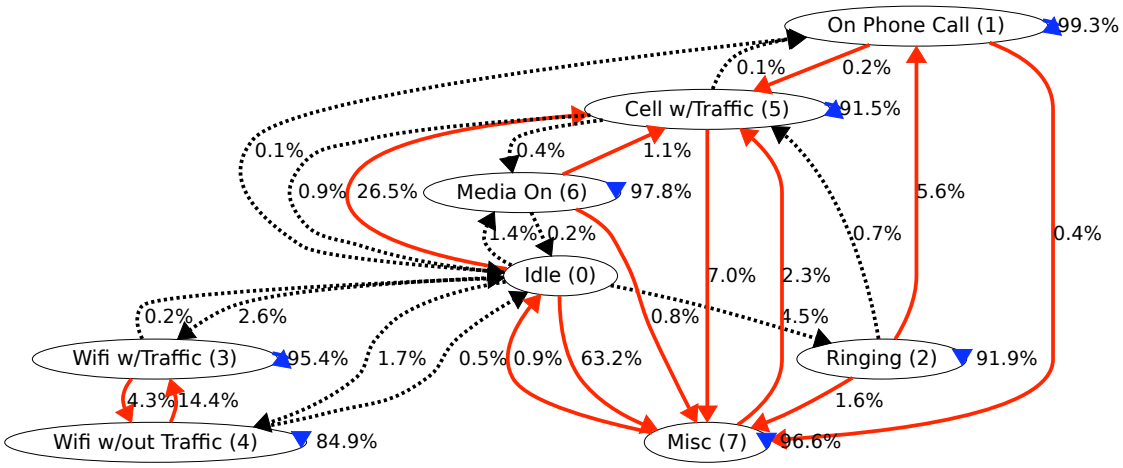

Figure 12: Common transitions between classified usage states.

Data in this form may be fed into any of multiple existing random state change prediction models.

One such state change prediction model is a Markov decision process (MDP). Applying the collected *Active* data to the MDP model results in the graph shown in Figure 12.

- Each node represents a state within the manually selected set of usage states, described in Table 7.
- Each edge represents a state transition; an arrow marks each tail state. The probability of following the transition labels each edge. Edges with a probability below 0.1% are not shown. Transitions from a node to itself (loopback) are shown in blue. For each node, edges with a 15% or higher probability of being the exit path (*excludes* loopback) from the node are solid and red. Other edges are dashed and black.
- A "dummy" state of *Idle* is inserted at the beginning of each *Active* usage session.
- "Misc." state can be thought of as a combination of a *transitional* state, on the path between other states, and a *general use* state, when the phone is *Active*, but neither communicating nor playing media. Further insight may be gained by dividing this state in a meaningful way, since it comprises nearly half (47%) of the *Active* time.

Consider a user associated with a wifi access point, whose activity is not causing significant wifi traffic. Such a user is in state (4), *Wifi w/out Traffic*. This occurs for 3.8% of the *Active* time (state 4 in the top of Table 7). The probability that a representative user's activity will cause significant traffic in the following second is roughly 14.4% (indicated by a red arrow from state (4) to state (3) in Figure 12).

The full utility of such a representation is well beyond the scope of what can be presented here; however, we would like to provide some example uses. Firstly, since user interaction determines state transitions, this type of model may be used to extrapolate information about the users' interactions with the smartphone. Second, such a probabilistic model may be used to construct *user-like* platform workloads for modeling and testing purposes. Finally, since a meaningful model can be constructed with a manageably small number of states and transitions, a compact representation of complex behavior patterns is possible.

7.2 Automatic State Detection

A set of states is manually selected to represent input data in the previous section. Depending on the number of variables, and the complexity of the state space, manually selecting a subset of meaningful states may not be trivial. For the set of selected variables, even if all values are treated as booleans, the set of potential states to select is $2^9 = 512$, and this number increases exponentially with the number of variables introduced. To demonstrate a method for *automatically* selecting representative usage states in cases where manual selection is impractical, a clustering algorithm is applied to the data.

The clustering algorithm included in the SimPoint 3 [15] program phase analysis suite is applied to the *Active* state data. SimPoint 3 is a software suite designed to automatically extract program phases from samples of executed code with low error (less than 10% error [15]). Specifically, SimPoint 3 uses a k-means clustering algorithm[10], guided by statistical analysis, to extract representative states, or *centroids*, for the input data. Of the large set of potential states, SimPoint selects 19 centroids, shown in the "Automatically Selected" section of Table 7 to cover the data set.

Comparing the two sections of Table 7 shows a strong similarity between the characterizing attributes chosen manually (top) and automatically (bottom). In some cases, a single manually-chosen state is represented by multiple automatically-chosen states (clusters 0, 14, and 17 all correspond to samples where the phone is transferring data via the cell network). Some comparisons:

- "Misc." is split across many centroids (4, 7, 9, 11, 15, and 16). This state in the manually selected set represents a large portion (nearly half) of the input data, which the clustering algorithm divided according to CPU utilization. A state transition model with correspondingly defined states might be used to model "bursts" of CPU activity on the order of, ex, seconds.
- Three centroids (8, 12, and 13) are not clearly represented by manually selected states; however, according to SimPoint, these clusters represent a small portion (2%) of the data.
- The phone ringing is not represented in the centroids. This is due to the small amount of time the ringing state is observed (0.6%).

Finding 10: From a large space of possible states, only a few significant states and transitions are required to meaningfully represent smartphone usage patterns.
Implication: Building a useful state-transition graph to model smartphone user behavior from a large dataset is tractable.

Finding 11: Automatic clustering of usage logs closely matches manually selected states of interest.
Implication: Meaningful states may be extracted automatically from input data in order to build a state-based model of user behavior.

8. RELATED WORK

There exist several benchmark suites for the embedded and mobile domain that target processor-oriented workloads [4, 7]. However, the real workload of mobile smartphone is more complex, and involves studying user activity.

There are several prior studies related to logging mobile devices. Phillips studies user activity for predicting when put a wireless mobile devices [12]. MyExperience [5] is a tracing infrastructure that incorporates explicit user feedback for studying high-level user actions. Demumiex studies real usage to breakdown commonly used applications, and percentage of time on the phone [3]. Shye [16] performs a power breakdown and develops the power estimation model used in this paper. Rahmati [13, 14] reports on a user study regarding the non-voice aspects of smartphone use.

There are several works that have that are similar to our work, or have come to similar conclusions. MyExperience [5] includes a battery study for estimating the remaining battery life. Rahmati [13] performed a similar study and provides suggestions on improving the human battery interface. Li finds that the operating system code is important on a mobile platform. Li also makes a case for the importance of the operating system [9] in embedded platforms. Balachandran characterizes the public WLAN environment [2] and also map session durations to a General Pareto Distribution. To the best of our knowledge, this is the first work that comprehensively studies the whole system and network activity for implications on optimization and modeling.

Numerous works have been published on the topic of network characterization. Henderson [8] observes trends in traffic on a university campus wireless network, showing varying phases over time. Yang [20] investigates general traffic self-similarity to motivate the use of a model based on power law. Our work models network traffic of mobile devices on both wifi and smartphone data networks, and notes characteristics specific to smartphone network activity.

9. CONCLUSION

In this paper, we study the smartphone usage of 25 users, representing a combined total of over 3 years worth of recorded use. To the best of our knowledge, we are the first to present a comprehensive analysis of the usage behavior, system performance, and network characteristics with respect to smartphone usage. We present 11 findings with implications on improving energy efficiency, modeling EDGE and wifi traffic, and automatically extracting usage patterns from trace data to develop model for user activity.

We must note that the data in this paper is from the Android G1 during the early stages of smartphone adoption, and may not represent the activity of other smartphones. However, there are likely to be similar characteristics across smartphones in the near future. In the long term, we believe that the design and optimization of future mobile architectures will benefit from subsequent studies across various settings and scenarios.

10. REFERENCES

[1] Arbitron and Edison Research Media. The Infinite Dial 2008: Radio's Digital Platforms.

[2] A. Balachandron, G. M. Voelker, P. Bahl, and P. V. Rangan. Characterizing user behavior and network performance in a public wireless lan. In *SIGMETRICS*, June 2002.

[3] R. Demumiex and P. Losquin. Gather customer's real usage on mobile phones. In *Mobile HCI*, pages 267–270, September 2005.

[4] EDN. Embedded microprocessor benchmark , http://www.eembc.org.

[5] J. Froehlich, M. Y. Chen, S. Consolvo, B. Harrison, and J. A. Landay. MyExperience: A system for in situ tracing and capturing of user feedback on mobile phones. In *MOBISYS*, 2007.

[6] Google, Inc. Android - An Open Handset Alliance Project. http://developer.android.com.

[7] M. R. Guthaus et al. Mibench: A free, commercially representative embedded benchmark suite. In *Workshop on Workload Characterization*, 2001.

[8] T. Henderson, D. Kotz, and I. Abyzov. The changing usage of a mature campus-wide wireless network. *Computer Networks*, 52(14):2690–2712, 2008.

[9] T. Li and L. K. John. Run-time modeling and estimation of operating system power consumption. In *SIGMETRICS*, 2003.

[10] J. MacQueen et al. Some methods for classification and analysis of multivariate observations. In *Proceedings of the Fifth Berkeley Symposium on Mathematical Statistics and Probability*, volume 1, pages 281–297, 1966.

[11] E. Mills. AT&T takes the phone out of iPhone, September 2009.

[12] C. Phillips, S. Singh, D. Sicker, and D. Grunwald. Applying models of user activity for dynamic power management in wireless devices. In *Mobile HCI*, September 2008.

[13] A. Rahmati, A. Qian, and L. Zhong. Understanding human-battery interaction on mobile phones. In *MobileHCI*, September 2007.

[14] A. Rahmati and L. Zhong. A longitudinal study of non-voice mobile phone usage by teens from an underserved urban community. Technical report.

[15] T. Sherwood, E. Perelman, G. Hamerly, and B. Calder. Automatically characterizing large scale program behavior. In *ASPLOS*, 2002.

[16] A. Shye, B. Scholbrock, and G. Memik. Into the wild: Studying real user activity patterns to guide power optimizations for mobile architectures. In *MICRO*, December 2009.

[17] L. Snol. More smartphones thank desktop pcs by 2011, 2009.

[18] Wikipedia: The Free Encyclopedia. HTC Dream. http://en.wikipedia.org/wiki/Gphone.

[19] J. Wortham. Customers Angered as iPhones Overload AT&T. *New York Times*. September 2, 2009. B1.

[20] C. Yang, S. Jiang, T. Zhou, B. Wang, and P. Zhou. Self-organized Criticality of Computer Network Traffic. In *Communications, Circuits and Systems Proceedings, Intl. Conf. on*, volume 3, 2006.