mStress: Supporting Continuous Collection of Objective and Subjective Measures of Psychosocial Stress on Mobile Devices

ABSTRACT

Stress can lead to significant health problems. However, development of new methods for better coping with stress have been challenging due to difficulty in capturing scientifically valid datasets from natural environments. Wearable sensors, which can capture physiological response to stress, are prone to noise and failure. In addition, aspects of everyday life (e.g., conversation, activity, etc.) confound the physiological responses to stress, making it difficult to tease out the effect of stress from changes in physiology. To overcome the challenges of assessing both exposures and responses to stressful events, new wireless sensing systems are needed.

In this paper, we present the design and evaluation of mStress, a smartphone-based (Android G1) system that continuously collects and processes measurements from six wearable sensors to infer in real-time whether the subject wearing the sensors is stressed. mStress generates prompts for timely collection of self-reports, triggered by real-time changes in stress level inferred by the system, to collect the subjective experience of stress when it is fresh in the participant's mind. To improve the quality of data, mStress incorporates several features including real-time detection of confounding events that affect physiological signals, and real-time detection of sensor detachments so the participant can restore connections. All of this functionality occurs locally on the mobile phone.

mStress has been used by 59 human volunteers in two scientific field studies, in which each participant wore the sensors and provided self-reports during their waking hours for 2-3 full days in their natural environment. The phone operated for 14 hours each day. Over 900 million samples of sensor measurements were collected and 65,000 stress predictions were made in 1,700 hours that mStress was used. mStress is being adopted in several new scientific user studies and facilitating the development and validation of inferring rich human behaviors (e.g., stress, conversation, etc.) from physiological measurements collected in the natural environment.

IPSN 2011 Chicago, IL USA

1. INTRODUCTION

Identifying effective methods to manage or cope with stress remains an open health challenge. Existing methods to measure stress rely on self-report or laboratory-based assessment with multiple methodological and logistical challenges related to their reliability, portability, burden on participants, susceptibility to multiple sources of errors and bias, and requirement of extensive human involvement from the participants and research staff. While the laboratory study lacks ecological validity, measures collected from the field such as self-report or biosamples (e.g., Cortisol from saliva or urine) can be burdensome, unreliable, biased, and episodic [34]. The consequence is a lack of rich, scientifically valid data about the experience of stress *in everyday life*. To capture such rich datasets, technological advances in mobile computing and wireless sensing are needed. Behavioral scientists and clinicians need mobile, ubobtrusive, high-quality, wireless sensing tools to robustly capture an individual's experience of stress in the field. With such data, behavioral scientists and clinicians could develop effective interventions to monitor and manage stressors that could lead to improved quality of life.

Collection of such rich data set from the natural environment involves several challenges. First, sensor noise, motion artifacts, and sensor failure make it difficult to reliably capture high-quality physiological indicators of stress (e.g., heart rate, sin temperature) in natural environments. Second, numerous everyday physical activities (e.g., walking) confound the measurements because they overwhelm the physiological signals and make it hard to filter out the changes induced by psychological stress. Third, there are wide between-person differences in the contextual, physiological, behavioral, and subjective factors that define the experience of stress [5, 1]. This creates a challenge in building a universal model that can be used to infer stress in real-time from physiological measurements. Consequently, to the best of our knowledge, there exists no smart phone based system that, in real-time, is able to capture a variety of complex contextual, physiological, and behavioral factors that define the experience of stress. Doing so requires reliably inferring stress from physiological measurements so that subjective measures (self-report) and other contextual factors can be collected close to the occurrence of stress event.

Motivated by these challenges, this paper proposes and evaluates **mStress**, a mobile phone (Android G1)-based system that supports real-time detection of stress. mStress collects continuous physiological measurements from six body-worn wireless sensors — Galvanic Skin Response (GSR),

Electrocardiogram (ECG), triaxial accelerometer, skin & ambient temperature, and respiratory inductive plethysmograph (RIP), all located on the chest. These measurements are processed into 14 unique features (e.g., heart rate variability). These features are fed to a support vector machine (SVM) to detect stress events that are personalized to each individual. mStress also collects self-reports on the subjective experience of stress and associated contextual factors. Reports are solicited when a user transitions in or out of a period of stress, as well as at fixed time intervals.

In addition, several measures are taken to maximize the quality of collected data and stress inferences. First, to motivate users to complete self-reports in a timely manner, users are paid micro-incentives for each completed self-report, and micro-incentives double if participants begin self-reports within seven minutes of a prompt for a report. Second, as the physiological response to stress is influenced by physical activity, a decision tree infers when the user is doing significant physical activity. This allows suppression of self-reports triggered by stress inferences when stress inferences are likely unreliable. Third, inferencing algorithms are used to detect if the user is wearing the sensors properly. If a sensor is not worn properly (e.g., an ECG electrode loosens from the skin), the system detects it in real-time and guides the user to correct the problem by presenting a series of instructions on the mobile phone. Fourth, to account for dominant confounding factors, the posture of the user and whether the user is speaking is detected using decision trees. These inferences are used as additional features to the SVM to contextualize the inference of stress.All of this functionality occurs entirely on the mobile phone without any help from the back-end cloud.

The primary reason for doing all computation on the mobile phone is to reduce the cost of running the study by eliminating the cost of cellular network communication and to simplify the logistics of managing multiple SIM cards and their plans. However, local computation has additional benefits. It reduces energy use, removes the need to build and maintain the backend, and reduces privacy risks to the participant, since no data is transmitted from the mobile phone to an off-site location outside the participant's control. We note that in the future, mStress can be extended to have connections to a back end server and the cloud.

The mobile phone lasts 14 hours on a 2300mAh battery when running the mStress application continuously. It takes less than 2 minutes to produce a stress inference, the majority of which is spent in computing various features. mStress was used to successfully conduct two real-life scientific studies in which 59 human volunteers wore the entire system for two or three full days in their natural environment. Over 900 million samples of sensory data were collected, 65,000 stress predictions were made, and 3000 prompts for self-report were answered, 98% of which were completed in less than 2 minutes.

Potential New Applications. Although mStress has primarily been used in scientific field studies for assessment of stress, it can be used to realize several new applications. First, real-time inferences of stress could be used to trigger **timely interventions** relevant to the user (e.g., a picture of a meaningful symbol, such as a family member, might appear on the phone, or the phone could play a "happy" song) when a user's stress level is too high. Second, **reactivity to an intervention** - how it changes physiology and stress levels - could also be measured in real-time. This would enable personalized selection/evaluation of interventions in the field. Third, common, everyday **interruptions** - a significant source of stress - could be managed by the phone based on the user's current stress level. For example, a call from one's boss might be routed to voice mail if previous measurements indicate a call from the boss when at home leads to excessive stress [8]. Last, but not least, stress measurements could also be used as part of a system that extracts and uses subjective information about a person from her sensor data (subjective sensing [21]). For example, real-time measurements of stress could be linked to a subjective navigation system which chooses a longer, but less stressful, route to work.

Organization. We discuss the requirements for the mStress system and the associated design challenges in Section 2. After briefly describing the wearable sensor system (in Section 4), we describe the mStress system. We discuss the mStress engine in Section 5, and the user interfaces in Section 6. The evaluation of mStress and experience from its use in a real-life scientific study appears in Section 8. The paper concludes with a discussion of related and future work, respectively, in Sections 9 and 10.

2. REQUIREMENTS

To meet the needs of the scientific community that studies stress, mStress must support reliable and timely data collection, data quality management, real-time context inferencing, ecological momentary assessment, and participant burden management.

Reliable and Timely Data Collection: The system must receive and store frequent measurements (10-60Hz) captured concurrently by multiple wireless sensors in real-time without noticeable losses. Measurements need to propagate through various layers in the system quickly so that timely inferences can be made about whether the user is stressed and if so, to collect other contextual information.

Data Quality Management: To meet the stringent data quality requirements of scientific studies, the system must provide the data quality controls of laboratory environments in the natural environment. For example, speaking affects some of the same physiological measurements that are affected by stress. In a lab environment, a study organizer can ask the user to remain quiet while measurements are taken. This is not possible in the natural environment. Thus, the system must infer when the user is speaking, and take this information into account when making real-time stress inferences to ensure its robustness.

Another problem which can be corrected easily in the lab environment but not in the natural environment is a failing sensor. For example, over the course of a day, an ECG electrode may gradually loosen from the user's body. This loosening results in a gradual degradation of ECG data quality over the course of day. In a controlled lab environment, a study organizer would monitor the quality of the data collected and correct any problems as they occur. Since a study organizer will not be able to continuously monitor and correct such sensor problems all day-long in the natural environment, the system must detect these problems automatically. When a sensor problem is detected, the system must inform and guide the user on how to correct it to ensure restoration of the quality of sensory measurements.

Real-Time Context Inferencing: To reduce unnecessary

energy consumption, an inference should only be made in real-time on the mobile phone if the inference is needed to trigger quick or timely action (by the system or user) in the natural environment. Additional data analysis or inferencing that is not needed in real-time in the natural environment can be done off-line after the study.

To enable taking timely action in natural environments, mStress needs to produce five different inferences, two of which are described above (sensor detachment and speaking). Third, significant, sudden increases or decreases in stress level need to be detected quickly so that users can be prompted to complete self-reports on stress and other contextual factors. The sooner a report is completed, the more accurate the report is likely to be. Fourth, since the effect of physical activity overwhelms the effect of stress on physiological measures, stress inferencing should be deactivated when intense physical activity is present. By suspending the computation of features and inferences associated with stress upon detection of physical activity, energy is not wasted on computing a likely incorrect stress inference. Fifth, the system must detect posture (sitting and standing). Posture has a similar effect on physiological signals as speaking [14], and is used (similar to speaking) as additional features to contextualize the inference of stress.

Ecological Momentary Assessment: Ecological momentary assessments (EMAs) are self-reports used to collect subjective data from study participants in the natural environment [31]. To enable personalization of the stress inferencing algorithm to each participant, the system must support collection of perceived stress from participants via EMAs that act as ground truth for the training and personalization of stress inferencing. In addition, the EMAs gather information about other contextual factors associated with stress that cannot easily be inferred in software. This data is critical to contextualizing stress - understanding what contextual factors trigger or are associated with stress. As described in the previous section, EMAs should be requested soon after a suspected stress event occurs to ensure high accuracy in participant responses. In addition, EMAs should also be requested on a timed interval, to enable capture of a baseline to which stress and contextual measurements can be compared.

Participant Burden Management: The burden of participation in an ambulatory study is often high. Participants must wear potentially uncomfortable wireless sensors on their body while going about their normal daily life. They must also deal with frequent requests for EMAs, correct problems with sensors, and replace or charge batteries. The system must take measures to reduce or mitigate this burden. Specifically, incentives should be used to motivate users to wear the sensors and complete EMAs. To further reduce the burden and improve compliance, users should also be able to specify a time period when they do not wish to be bothered by EMAs (e.g., during an exam or while sleeping).

Additionally, to allow conducting a study with untrained users, the system must make participation as easy as possible. For example, participants should be able to easily and quickly validate correct operation without the need for complex troubleshooting.

3. DESIGN CHALLENGES

Meeting the requirements presented above presents several design challenges for mStress, some of which are discussed below.

Feasibility: Real-time inferencing on mobile devices has been demonstrated in some cases, such as using accelerometers to classify physical activity [7]. However, the real-time detection of richer psychological events such as stress response has not been demonstrated previously on a mobile phone.

Energy Efficiency: To capture the data needed, mStress must be able to run on the phone uninterrupted for the length of a field study (at least several days). However, the real-time inferencing pipeline requires significant energy resources to buffer samples from multiple sensors, compute features from the samples, and process and fuse features to produce inferences. One could provide the participant with additional batteries or chargers. However, frequent battery changes or recharging sessions (e.g., once every 6-24 hours) introduces gaps in data collection, as well as increases the burden of participating in the study [9]. Thus, it is necessary to optimize the system's energy consumption. The simplest solution is to reduce the frequency of sampling, feature production, and inferencing. However, stressful events can occur in an instant (e.g., sudden breaking while driving). Important stress events may be missed if the frequency of inferencing is too low.

4. WEARABLE SENSOR SYSTEM

The mStress system uses wearable sensors (Figure 1) to monitor cardiovascular, respiratory, and thermoregulatory systems, systems known to respond to stress and other psychologically and physically demanding conditions. Six sensors were chosen: 1) an electrocardiogram (ECG) attached to the body with two electrodes to measure electrical output of the heart, 2) a sensor measuring skin conductance between the two ECG electrodes, 3) a skin temperature thermistor attached to the skin, 4) an ambient temperature sensor, 5) a three-axis acceloremeter, and 6) a respiratory inductive plethysmograph (RIP) band to measure relative lung volume and breathing rate. The sensors are implemented on two wireless (802.15.4) motes. One mote is dedicated for the RIP sensor and the second mote implements the remaining modalities, enabling the study coordinators to exclude the RIP band if respiration features are not required in their particular study. Each mote is 2.5 square-inches and powered by rechargeable 750 mAh batteries. The lifetime for the streaming mode is up to 72 hours for moderate datarate (60 samples/node/sec). The system also uses an 802.15.4-to-Bluetooth bridge that captures packets of samples sent by the sensor motes and sends them to the phone via Bluetooth. More details on the wearable sensor system will be reported separately.

5. MSTRESS ENGINE

The mStress engine uses the pipe and filter architecture concept [4, 29]. Data is passed through four system layers (filters) to produce inferences as can be seen in Figure 2. First, data from the sensor motes arrives at the engine's **Network Layer**, where it is packetized and demultiplexed. Additional sensor data may also be read from the phone's internal sensors (e.g., GPS, accelerometer). Second, sensor data is passed to the **Abstract Sensor Layer**, where data from each sensor is added to a sensor buffer, an abstraction of a sensor that buffers sensor data into windows. Third,

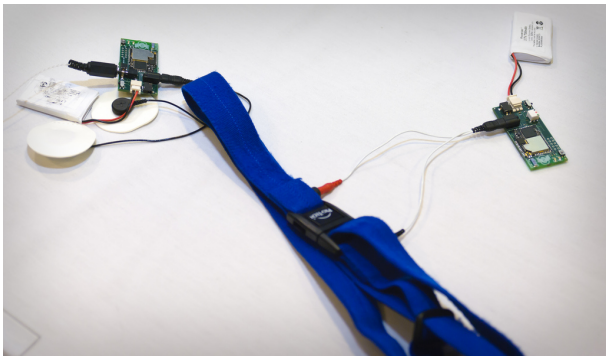

Figure 1: Wearable sensor suite used by mStress. The suite includes six sensing modalities: electrocardiogram, skin conductance, skin temperature, ambient temperature, three-axis accelerometer, and a respiratory inductive plethysmograph band.

when a window is complete, the **Features Layer** computes features from the window. Two separate components in the Features Layer compute features, **virtual sensors** and the **feature statistics** module. Virtual sensors process a window of sensor data to produce a new window of virtual sensor data (e.g., a virtual sensor could produce a window of R-peak locations from a window of ECG data). The feature statistics modules computes statistics, such as mean, variance, heart rate, and respiration rate. Fourth, once features are computed, they are then passed on to the **Inferencing Layer**, where inferences are computed from the features.

Communication between these layers is provided by a series of buses (pipes) that follow the Observer design pattern [12]: a Mote Bus that passes mote sensor data from the Network Layer to the Sensor Layer; a Sensor Bus that passes windows from sensor buffers (including virtual sensor buffers) to the Features Layer; a Feature Bus that passes feature statistics from the Features Layer to the Inferencing Layer; and a Context Bus that passes context inferences to the user interface. A logger listens on all the buses, and logs all sensor, feature, and context data for offline post-processing and validation.

5.1 Network Layer

The network layer receives data from the Bridge Mote by Bluetooth and decodes it for distribution to the various sensor buffers that drive the mStress Engine. In addition, it also plays the role of a transport layer, where connections are started, stopped and managed. The key tasks of the network layer are *connection management*, *data reception and decoding*, *demultiplexing data from sensors*.

Connection Manager: The Connection Manager establishes and tears down connections to the Bridge Mote, reads and writes raw bytes to and from the Bridge, and monitors connection and mote health. The Connection Manager can notify the application of various error conditions such as if the connection to the bridge is lost, if the bridge is moving out of range, or if it is not receiving data from a specific mote. If the connection to the Bridge Mote has been lost, the Connection Manager can search periodically for the Bridge Mote and reconnect automatically when it is back in range.

Data Reception and Decoding: Raw bytes coming in from the Bluetooth connection are locally buffered in a block-

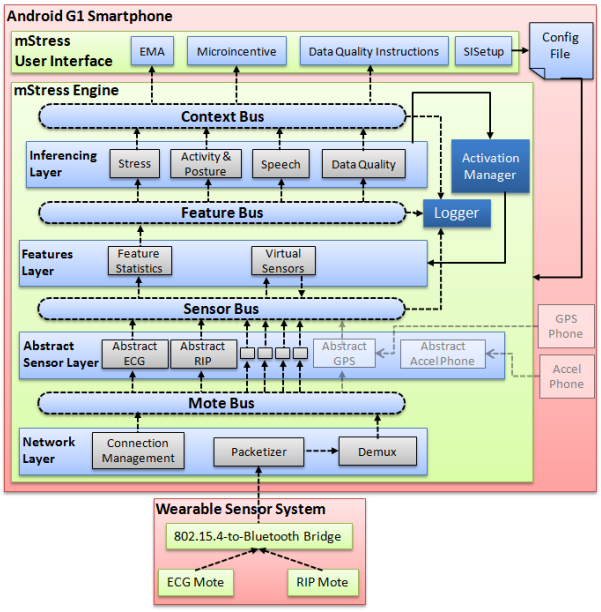

Figure 2: The mStress system, including the physical sensors, engine (Network, Abstract Sensor, Features, and Inferencing Layers), and user interfaces, as well as the Mote, Sensor, Feature, and Context communication buses. Dotted arrows represent the flow of data through the system, and solid arrows represent commands (e.g. activation/deactivation and configuration). Some sensors are faded to denote that they were not used in the deployed system, but could be used if needed.

ing queue. A Packetizer reads this buffer and organizes the received bytes into valid tinyOS packets. The Bridge Mote relays data from the Sensor Motes to the phone using the tinyOS serial protocol, PPP in HDLC-like framing [15]. Thus, the Packetizer can use the standard PPP error and synchronization checks such as CRC, FCS and Flag sequence to check for errors in synchronization and data corruption.

Demultiplexing: The demultiplexing phase establishes a link between a physical sensor on the body and an abstract sensor, an abstraction of the sensor on the phone. Once a packet is produced by the Packetizer, it is demultiplexed and distributed to the appropriate abstract sensor via the Mote Bus. Similar to the way TCP distributes data by a port number and an IP address, packets from a specific analog-to-digital converter (ADC) channel on a specific mote are mapped to a corresponding abstract sensor.

5.2 Abstract Sensor Layer

The Abstract Sensor Layer provides two functionalities to the mStress engine. First, it facilitates code reuse by providing a common interface to all sensor data, independent of the source or type of the sensor. Each physical sensor - whether on a mote or on the phone - has a corresponding abstract sensor in the engine, which abstracts away the source of the data. Abstracting the source of the data enables using one implementation of a feature computation algorithm on windows from many different sensor types or sources.

Second, the Abstract Sensor Layer buffers sensor data into windows from which features can be computed. Using a

Strategy design pattern [12], a different windowing strategy can be selected based on the particular use case. For example, all physiological sensors were implemented with a fixed window size strategy, where a fixed number of samples is aggregated into a window. However, a fixed window size strategy would not be appropriate for event-driven sensors, where new samples are only issued when specific events occur (e.g., accelerometer or GPS on a phone). If fixed window sizes are used with event-driven sensor data, then the data could potentially sit in an unsent window for a long time if new events do not occur often. A timer-based windowing strategy could be used to send a window at a reasonable frequency. When a fixed-size window is full or a timer elapses for timer-based windows, then the window of abstracted sensor data is sent to the Features Layer via the Sensor Bus.

5.3 Features Layer

The Features Layer is responsible for computing features that will be used to make inferences. Two types of feature data are computed in the Features Layer, statistical features and virtual sensor data.

5.3.1 Activation Manager and Feature Addressing

Given the limited computational and energy resources of the mobile phone, the engine incorporates an activation manager that ensures that a feature is computed only when an active inferencing implementation requires that feature. This requires an addressing scheme that allows inferencing modules to specify which features they need. Our addressing scheme was developed based on three observations. First, we observed that several features used by the inferencing implementations are computed over more than one type of sensor data. Second, each inferencing implementation needs different features computed on different sensors. Third, the same features were being used by multiple inferencing algoriths. Given these observations, a feature addressing scheme was designed that uniquely encodes both the feature to be computed and the sensor data the feature will be computed from in a single five-digit integer address. The three higher-order digits of the address correspond to the feature to be computed and the remaining two lower-order digits correspond to the sensor that features should be computed from. Simple helper functions and a set of human-readable constants are used to simplify the process of coding and decoding feature-sensor addresses. For example, if the classifier Model1 needs the mean (ID 101) of the accelerometer magnitude (ID 14), then Model1 would construct the address as follows:

```
 Constants.getFeatureSensorID(
   Constants.F_Mean,
   Constants.S_Phone_Accel_Magnitude)
//   return 10114
```

The activation manager ensures a feature and sensor are only activated once when at least one request for the same feature-sensor pair occur, and only deactivated when none of the active inferencing implementations require it. Combined with the Feature Bus, this enables multiple inferencing algorithms to share the computation of this feature rather than compute it once for each inferencing implementation that requires it. A Factory [12] is used to load the correct sensor and feature objects at runtime, allowing for easy addition of the features or sensors to the engine. An alternative to this addressing scheme would directly pass the feature and sensor object references rather than integer addresses. However, this would require constant lookup in object lists, an expensive operation in Android.

5.3.2 Feature Statistics

The Feature Statistics Module computes over 30 features from windows that are sent to it via the Sensor Bus. When a new window arrives, the sensor associated with the window is checked against the Activation Manager's list of active sensor-feature pairs representing features that should be computed. All matching sensor-feature pairs are then added to a queue as pending feature computation jobs. A separate thread removes jobs from the queue and processes them. Using a separate thread prevents feature computation from blocking other computation from occurring in the engine. However, we chose not to compute multiple features concurrently as the repeated creation and destruction of 30 or more threads (1 per feature) significantly increased computational delay on the mobile phone.

Features computed range from simple statistics such as mean and variance to respiration amplitude and heart rate. All features were implemented using known methods. Where possible, lighter weight implementations were used to reduce CPU usage and preserve battery life. For example, heart rate variability was computed in the frequency domain using the Lomb periodogram [6]. Lomb periodogram is a less expensive approach because, unlike a tachogram-based approach, the Lomb periodogram does not require interpolation. It also does not require evenly sampled data, an important consideration given that some packets arriving from the motes could be lost. When a job is completed, its result is put on the Feature Bus.

5.3.3 Virtual Sensors

Some features are derived from a common set of intermediate features which are expensive to compute. To avoid repeatedly computing these expensive intermediate features, we introduce the concept of Virtual Sensors. Virtual Sensors compute intermediate features from windows of virtual or abstract sensor data. They are called virtual sensors because, similar to abstract sensors, they produce windows of virtual sensor data - features - and place these windows on the Sensor Bus. For example, the detection of R-peaks (beat-to-beat intervals) in the ECG signal is implemented in a virtual sensor. Ten ECG-related features used by the stress inferencing algorithm are derived from R-peaks. Thus, implementing R-peak computation in a virtual sensor buffer means that the R-peaks are only computed once in a virtual sensor rather than ten times (once per features), an order of magnitude savings.

5.4 Inferencing Layer

Using feature statistics that arrive over the feature bus, the Inferencing Layer executes machine learning algorithms to produce inferences about the state or context of the participant. All inferencing implementations internally cache the features they receive from the Feature Bus. Once all needed features arrive, the inferencing implementations produce an inference and put the result on the context bus. If an inferencing implementation uses online training, then a context label manager passes new labels from the user interface (mainly EMA) to the underlying machine learning models for retraining.

Four types of inferences are made on the phone, personalized stress, posture and activity, whether the user is speaking, and sensor detachments.

5.4.1 Personalized Stress Inferencing

The personalized stress inferencing algorithm uses Support Vector Machines (SVMs) [30] to produce binary inferences indicating whether the user is stressed (stressed/not stressed). Although SVMs are state-of-the-art machine learning techniques and have been shown to yield excellent performance in many applications, their generalizability might be limited if the inter-subject variation is large, which is the case with physiological response to psychological stress [14, 5]. We therefore incorporate person-specific information into the stress detection model to overcome this challenge. The SVM was trained both offline and online (on the phone) from participant perceptions of their stress level (provided via EMA). Offline training data was collected from participants in a lab session and from a day in their natural environment. The offline training data was used to bootstrap an online training algorithm that was used during a second day participants spent in their natural environment.

For implementation on the phone, the personalized stress inferencing algorithm requests 14 unique features computed over ECG and GSR from the state manager. Remaining features were found to be too noisy or not discriminative enough for stress. As ten of the ECG features were computed from R-Peaks, an R-peak detector was implemented in a virtual sensor. This reduced computation of R-Peaks to once per window rather than once for each ECG feature. Once all features arrive, they are passed to libsvm, a java-based library for training and producing inferences from an SVM.

When a participant responds to an EMA requesting information about their stress level, the responses are passed to the personalized stress implementation via the context label manager. Once a new label arrives, the personalized stress implementation retrains the model if needed. In general, retraining an SVM is an expensive operation. Given the processing and battery limitations of the smart phone, it was necessary to restrict the initial model (computed offline) to have 300 support vectors. With this smaller SVM, retraining currently takes only 20-30 seconds. Our results for 20 subjects indicate that the average recall for online learning is 79% and precision is 84%. Details on the design of the stress inferencing algorithm will be reported separately.

5.4.2 Posture and Activity Inferencing

Activity inferencing utilizes accelerometer data from the wearable sensors to identify if the user is sitting, standing, and walking. Accelerometers have shown to be highly effective in activity recognition when combined with machine learning algorithms [3, 28]. The inferencing was implemented using a decision tree, as previous work indicates that a decision tree provides a good balance between accuracy and computational complexity [23].

Implementing posture and activity inferencing in the mStress engine involved requesting appropriate feature activation and processing of incoming features with the decision tree. The implementation uses two features, mean adjusted deviation and mean crossing rate of the Z accelerometer (perpendicular to the body, facing forward). These two features are used to traverse a decision tree trained offline. The decision tree had a 90% accuracy rate in leave one subject out cross validation of training data.

5.4.3 Detection of Speaking Events

We infer whether the subject is speaking using features derived from the respiration signal (chest volume sampled at 60 Hz). Definition of the features are based on the proper identification of a respiration cycle, which is composed of an inhalation period followed by an exhalation period. Various statistics (e.g., mean, median, and standard deviation) across five respiratory cycles are computed for three features — inhalation duration, exhalation duration, and their ratio, called IE ratio [24]. We train a decision tree from offline data (that selects 4 features) and obtain accuracies of 93.08%, with Kappa = 0.8612 using 10-fold cross validation.

Accurate identification of peaks and valleys in a respiratory cycle is required to compute all the features. To avoid recomputing the peaks and valleys for each feature, we use a virtual sensor implementation for this calculation. The "real peak-valley virtual sensor" takes a window of 1800 raw respiration samples as input, and produces a window of indices of the real peaks and valleys in the input window. A second stage of virtual sensors, the inhalation, exhalation, and IE-ratio virtual sensors, listens on the sensor bus and computes windows of inhalation duration, exhalation duration, and IE-ratio from the peak-valley pairs, and puts the result back on the Sensor Bus. The Feature Statistics modules listens on the bus for these windows and computes the appropriate statistics from these windows. These statistics are then passed on to the inference module.

5.4.4 Detection of Sensor Detachments

Currently, four of the six sensors are monitored for detachment — ECG and GSR via gel-filled electrodes, respiration via a band that goes around the chest, and temperature via a probe affixed with an adhesive. The detachment or drying of electrode may produce excessive noise, low signal amplitude, or even saturation of the signal [10]. The instruction in each case to the participant is to replace the electrode. To detect excessive noise and saturation, raw ADC measurements of ECG electrode are used, whereas for detecting low signal amplitude, the variation in R-to-R intervals (produced by a virtual sensor) is used. Loosening of the RIP band produces excessive noise, whereas detachment of the connector produces saturation. Raw ADC measurements are again used to detect saturation, while the *stretch* (difference between successive peak and valley in a respiratory cycle) produced by a virtual sensor is used to detect loosening of the band. For both cases, the action recommended to the participant is to tighten the band. For temperature probe detachment, raw measurements are checked for outside a dynamically computed range. The recommended action is to press the adhesive.

5.5 Energy Management

There are several scenarios where specific inferences and the computation of the features they require should be deactivated by the activation manager to save energy. For example, stress, speaking, and sensor detachment inferencing should be deactivated when the participant is undergoing significant movement (e.g., moving in a chair, walking, and exercising). During significant movement, the physiological markers of stress, speaking, and sensor detachment are over-

whelmed and their inferences are not reliable. In addition, the detachment of a sensor should trigger deactivation of any inference that depends on that sensor. If the detachment is accidental, the user can correct the problem in a short period of time, and little energy savings would be expected. However, if the user detaches the sensors on purpose for a significant period of time (e.g., taking off the sensors before taking a nap or going for a jog), then significant energy savings could occur.

5.6 Implementation on Android G1

We implemented mStress on an Android G1 mobile phone. Although newer, more powerful smart phones are currently available, the challenges encountered on implementing the software on the G1 will likely remain even with newer phones since future mobile sensing and inferencing systems will require more power. The number of sensors will grow to more than 10 and features to more than 100 and the number of inferences will increase as well.

Challenges arose with peculiarities of the Android platform. Object creation and garbage collection are expensive on Android. Following the advice in the Android Designing for Performance guide [13], object creation was carefully limited throughout the engine. Wherever possible, scalar primitive types are used. For example, all abstract sensors buffer sensor data as primitive integers. We also chose to implement the entire functionality of the engine within a single service. This reduces the need for inter-process communication, another expensive operation on Android. Putting the engine within a single service also allows the engine to run continuously without regards to the user interface.

In almost all cases, we implemented functionality using the Java API rather than native code. While this increases memory use and computational delay, software development spread across three universities was generally easier in Java and we found that, with one exception, we could compute over 30 features and make up to 4 inferences in real time. The one exception is the implementation of the respiration rate feature. An initial Java implementation took several minutes to complete but, on average, took 30 seconds to complete after porting the code to native C. Native code was also used for Bluetooth connection management code, but only because no Bluetooth API existed for Android 1.6 when mStress was initially developed.

6. MSTRESS USER INTERFACE

mStress has user interfaces for both the study coordinator and the participant. We discuss their details in the following.

6.1 Study Coordinator User Interfaces

The user interfaces for the coordinator enable them to set the system up, personalize it to each participant, and to verify that the sensors and the mStress program on the phone are working.

SISetup: The study coordinator uses the SISetup interface to set up the mStress system for use in the field. In SISetup, the study coordinator can scan for and select the bridge mStress will connect to, select the participant's personalized stress detection model that will be used to trigger EMAs, and select time periods when EMAs should not appear. SISetup is hidden from the participant so that they do not change any settings during the study. The SIStudy interface starts the mStress program and displays an information screen confirming that data collection has started, as well as a reminder of the dead periods.

Verification: The mStress system needs to collect data from all sensors and EMAs. Since the mStress software is updated for each participant (to use the stress inference engine personalized to this subject), software errors may be introduced inadvertently. LEDs on the Android G1 are used as initial indicators of a problem with reception of sensory measurements. The LEDs blink red and blue as samples arrive at the phone from the sensors. Red blinks indicate reception of a packet from the ECG mote and blue blinks indicate reception from the RIP mote. Motes are replaced if a problem is observed. Study coordinators, who are not technical people, found this visual method of checking mote problems very convenient.

In the middle of the study, the component that generates EMA prompts stopped generating prompts for several hours. This compromised the data by reducing the number of EMAs collected. The system now uses two feedbacks to provide the study organizer with confidence that EMA collection is working correctly. First, the EMA module emits a tone and displays a prompt on the display once it is activated. Second, an EMA is scheduled for one minute after the system is started. This guarantees that the study coordinator will be in the room when the first EMA appears. If an EMA does not appear within a minute, there is a problem with the system, in which case technical member of the team is contacted to correct the problem.

6.2 Participant User Interfaces

Participants primarily interact with the mStress system via its user interfaces. There are four different interfaces that participants use in mStress — 1.) EMA interface to provide self-reports, 2.) microincentive feedback to encourage compliance, 3.) visual inspection for sensory data reception, and 4.) instructions to rectify sensor detachments.

Ecological Momentary Assessment (EMA). EMA is used to collect self-report data in a way that attempts to avoid sources of error, such as recall bias [31], that are introduced in more traditional diary, or interview methods. mStress uses EMA to obtain additional information about the user's context that can aid in labeling of the physiological measurements. The EMA questionnaire employed in mStress contains around a dozen questions. The first set of questions are ones that have been shown to be correlated with stress level as measured by blood pressure and/or salivary cortisol [18, 17]. The remaining questions allow us to collect ground truth information on non-physiological context such as location, and social interactions. The design of the EMA interface is quite simple containing a question at the top of the screen, a list of responses to that question and two control buttons at the bottom of the screen.

The user is prompted to complete an EMA questionnaire under one of two conditions. Either a sufficient amount of time has elapsed since the last completed EMA, in our case around an hour, or there has been a change in the output of the stress inference classifier. Timed EMAs are used to ensure that we collect sufficient self-report data throughout a participant's day in order to label data for offline analysis. The context-triggered EMAs are to ensure that we can obtain labels for periods of interest. In order to avoid undue burden on the participants and negatively impact their stress level, the EMA scheduler inserts a dead period of 30 minutes

immediately after each EMA. This ensures that the participant in not inundated with multiple EMA prompts within a short period of time. Additionally, the EMA scheduler ensures that the total number of EMAs (timed and stress-triggered ones) does not exceed 20 per day.

Although event-triggered EMA have been envisioned and used earlier, primarily in the context of physical activity or GPS assessment [16, 34], to the best of our knowledge, mStress is the first system to use rich psychological events such as stress to trigger EMAs.

Microincentive Feedback. As described previously, the incentives paid to the participant are based on timely response to EMA prompts and on the amount of time the participant wears the sensors. Before starting an EMA, the participant is briefly shown a summary of the microincentives earned so far. In addition to encouraging compliance, visual display of EMA also introduces transparency, so participants can verify that they are being compensated appropriately. The microincentives are logged upon every increment, so that if the mStress application crashes and needs to be restarted, the microincentives earned are not lost. Additionally, the microincentive unit must ensure that the incentives earned do not exceed the maximum compensation budgeted for each participant. This could occur if the number of EMAs exceed 20 each day. The EMA module ensures that this limit is not exceeded. An additional challenge that needs to be addressed is the effect of sensor detachments on microincentives since the incentives earned are partially tied to having worn the sensors in the period preceding an EMA.

Visual Inspection for Sensory Data Reception. A requirement of the mStress system is to maximize collection of physiological and self-report data that meets the stringent data quality requirements of scientific studies. Meeting this requirement is particularly difficult because the study coordinator cannot monitor the sensors and correct problems when the participant is in the natural environment. Sensory measurements may be lost for several reasons such as sensor detachments, bridge being out of range from sensors or from the phone, and software issues on the phone. The same LED blinking mechanism that is used by the study coordinator to verify data reception at the time of setup is used by the participants as well.

To minimize data loss, participants were instructed to check the LEDs every time they were prompted for an EMA (at least every 55 minutes), if not more frequently. If users noticed a problem, they could then take several steps to correct it. If both LEDs were not blinking, users were instructed to quit and restart the program. If the blinking reappeared after restarting the program, then the data reception problem was likely due to accidental disconnection from the bridge. Otherwise, data reception was failing because the RIP and ECG motes were too far away from the bridge, or the batteries on the motes were depleted. If only one phone LED was blinking, this indicated the bridge was working properly, but it was only receiving data from one of the sensor motes. In this case, the mote for which data was not arriving at the phone had depleted batteries or had moved too far from the bridge. If the RIP and ECG motes were far away from the bridge, the LEDs on the phone would blink again when moved closer together. Motes with depleted batteries were identified by checking if indicator LEDs on the motes were blinking. Participants were given extra mote batteries so they could change them in the field.

Rectifying Sensor Detachments. When a sensor detachment was detected by the system (see Section 5.4.4), the phone vibrated with a distinctive pattern to notify the user. The system also displayed instructions so that the participant could correct the problem. When the participant completed the instructions, the system checked if the problem had been corrected. If the problem had not been corrected, the user could go through the instructions again, or call the study coordinator, whose number appeared on the screen, for additional assistance.

7. VISUALIZATION OF MSTRESS DATASET

We developed a web-based visualization system for reviewing data collected by the mStress framework. The system processes the collected sensor data and inferences to produce four visualizations. The **Day at a Glance** graph depicts an overview of behaviors performed by the user in his/her daily life. For example, the visualization presents the fraction of time spent walking or in conversation. Stress is depicted here as durations of stress in a day (Figure 3). The **Stress at a Glance** and **Stress at Places** visualizations provide a more detailed view of stress. The former depicts the fraction of time the user was stressed during various behaviors. The latter depicts the fraction of time at places, such as home and work (gathered from self-report), when the user was stressed. Lastly, the **Daily Timeline** (Figure 4) visualization plots behaviors on a detailed timeline.

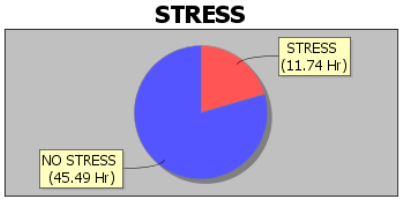

Figure 3: Fraction of monitoring period participant was stressed

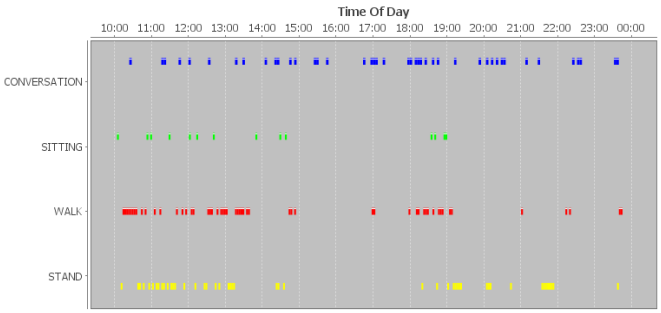

Figure 4: Daily Timeline of a participant monitored by mStress

8. EVALUATION & DEPLOYMENT EXPERIENCE

We evaluate the mStress system from two perspectives. First, we measure the energy profile of the deployed system, and compare it to other possible configurations of the system

that we chose not to use. Second, to characterize the timeliness of the system's stress inferences, we measure the computational delay introduced at each stage in the inferencing pipeline. Next, we present various interesting statistics of physiological, self-report, and behavioral inference that was collected by mStress from 59 participants who used mStress for 2-3 full days in their natural environment.

8.1 Energy Profile

We characterize the energy profile of the deployed version of mStress (condition SI). We measure the impact on the lifetime of the phone, if network communication (WiFi) to a backend server or cloud (condition SI+Network) were to be used, and if GPS coordinates were to be logged (condition SI+Network+GPS). Device lifetime is measured by running mStress continuously, starting with a full 1150mAh phone battery and ending when the battery reached 15% of its total energy capacity[1]. The wearable sensors actively transmitted samples throughout the test period, and a best effort was made to complete EMAs when they appeared.

The results appear in Figure 5. We observe that with the regular battery, the phone lasts only 8 hours (to deplete 85% of the battery) when running mStress. This was insufficient for capturing the wake hours of participants, which span more than 12 hours. Consequently, we switched to a larger 2300mAh battery. With this battery (condition SI+LargeBattery), the phone lasted over 13 hours to reach 15%. This measurement is consistent with the lifetime reported by the participants. On average, participants in the field reported wearing the system for 14 hours, and logs indicate approximately 12 hours of data from sensors was collected from each participant. The two hour difference is due to restarts and occasional disconnections.

We next observe that using the network connection reduces the lifetime by 26.5% (from 8.07 hours to 5.12 hours). Consequently, even if we use the larger battery, the phone would last only 8.36 hours to get to 15% level, which will be insufficient. Finally, we observe that logging GPS coordinates leads to an additional reduction of 13.7% in the lifetime. If the participants were to talk on the study phone and use the phone for emails, browsing, games, etc. then significant extension to the lifetime would be needed. Strategies proposed in [20, 33] for saving energy with GPS logging can be used, but this still leaves significant deficit if connectivity to the back-end is to be used, and if the set of wearable sensors and context inferences is expanded further. For example, from our current measurements, we have found that talking on the phone increases the current draw by 50mA (as compared to an average current draw of 121mA for the SI condition). Similarly, turning the display on increases the current draw by 90mA.

8.2 Computation and Communication Delay

We analyze the computation and communication delay of the system to verify that inferences are made in a timely fashion. Since stress inferences are used to trigger EMAs, it is important that EMAs appear near in time to the occurrence of the stress event being detected. The time between capturing a sample from the participant and the making of an inference is, on average, 118 seconds, or just under 2 minutes. Approximately 60 seconds of that time is spent buffering sensor data (1 minute windows). Approximately (approximately 40 seconds) is spent in the Features Layer computing statistical features. This delay is mostly attributable to queueing feature computation jobs and using a single thread to operate on them one-by-one. The Network and Abstract Sensor Layer introduce less than one second of delay. The remaining 18 seconds of delay comes from making stress inferences in the Inferencing Layer. Using buses for communication between layers adds virtually no delay. We note that significant delays can also be introduced at the Abstract Sensor Layer if packets stop arriving from the motes (e.g., because of a dead battery). If the system were to wait until a window of physiological data was full, partially filled buffers may never be sent. A timer-based window flush strategy as described in Section 5.2 is applied to send partially filled windows of physiological sensors, after sufficient time has elapsed since this window was started.

[1]Note that Android warns the user about the battery dying when it reaches 15%

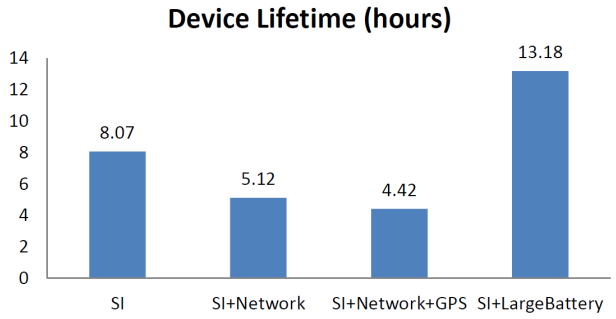

Figure 5: The lifetime of the mobile phone under various configurations of mStress.

8.3 Experience Report

mStress was used in two field studies (N=59 participants). Each study had a real-life behavioral science, human-computer interaction, or ubiquitous and mobile computing goal, and thus are candidate scenarios in which to test the viability of mStress for use in scientific field studies. We present deployment experiences for both studies together below. Results associated with the goals of the studies are reported separately.

8.3.1 Study Descriptions

Study I: The goal of the first study was to capture subjective and physiological responses to stress in natural environments. Twenty three participants from an 11,000+ student university wore the sensors during the awake hours of two separate days. Participants were instructed to go about their normal daily lives. To ensure the capture of subjective and physiological responses to stress, the first day was scheduled to coincide with one of the participant's exams - a likely stressor. One month later, participants returned for the second day of field study. The data gathered from the first day was used to train the stress inferencing algorithm, allowing capture of stress on the second day without a designated stressor.

Study II: The goals of the second study were to examine the use of micro-incentives for scientific data collection, identify the effect of interruptions on user stress level, and study participant concerns regarding privacy of continuously-collected behavioral data. Thirty six participants from a 20,000+ stu-

dent university wore the system during the awake hours of three continuous days. Again, participants were instructed to go about their normal daily lives while wearing the sensors. As the system did not last more than the awake hours of a single day, participants returned to the lab each morning to exchange depleted mote batteries for new ones. They were also given a charger for the mobile phone so that they could charge the phone overnight.

Improvements in mStress from Study I to Study II: Several changes were made to mStress (from the version used in Study I) to meet Study II's goals as well as address problems encountered in Study I. In Study I, the mobile phone sometimes disconnected from the bridge, leading to significant data loss (when disconnected, the bridge could not relay sensor data to the phone.) mStress was made more robust to disconnections by automatically trying to reconnect to the bridge on a timed interval. This reduced the burden on participants significantly, as they no longer needed to monitor the system for disconnections and initiate reconnections. Second, an oscilloscope was added to mStress to allow the study coordinator to observe sensor waveforms as they arrived at the phone. The oscilloscope was used to verify the sensors were attached properly and sensor data was received by the phone before sending participants into the field. Third, inference-based EMA triggering was generalized to work with any contextual inference produced by the system (not just stress). This allowed studying how stress induced by interruptions is mediated by context (reported separately). Fourth, event budgets were imposed on self-reports to ensure frequent events (speaking) do not trigger too many self-reports. Fifth, a generalized framework for incentives was added to mStress to allow testing the effectiveness of a variety of incentive schemes, which could be customized to each subject.

8.3.2 Robust Data Collection

Overall, data collection statistics show mStress was able to reliably and robustly capture sensor data in natural environments (Tables 1). Over 1,700 hours of data were collected from 59 participants, an average of 11 to 12 hours of sensor data per participant per day. Furthermore, sensor detachment detection played a role in minimizing loss of data. The percent of this data lost due to sensor degradation or detachment was only 7% in Study I. In Study II, data loss due to sensor degradation rose to 16%. This rise in data loss occurred because the data quality module was omitted in Study II, and participants were never warned to fix sensor detachments. Thus, the incorporation of online data quality detection and management is critical to maximizing high-quality data collection.

	Samples	Hours	% Degraded
Study I	140 million	270	7
Study II	200 million	400	16

Table 1: Average amount of sensor data collected per day of each study stated in number of samples and hours. Only a small percentage of the data was identified as degraded due to sensor detachment.

Data collection statistics also show mStress was able to reliably and robustly capture self-reports from participants in the natural environments (Tables 2). On average, participants in both studies completed 80+% of the self-reports requested by the system. This is significant because the number of self-report requests, each of which represents an interruption to the subject, were significantly higher than typically used in scientific studies (i.e., 20+ vs. 5). Furthermore, on average, participants in both studies started self-reports in 44 seconds or less, more than close enough in time to the event of interest to meet scientific standards. However, participants in Study I started self-reports faster than those in Study II. This can be attributed to the bonus incentive Study I participants received for starting self-reports within seven minutes. This bonus incentive was not used in Study II. Participants in Study I also completed self-reports faster, a result of using a longer self-report questionnaire in Study II.

Inferences were used to trigger a subset of the self-reports. On the first day of Study I, 25% of self-reports were triggered by stress inferences versus 65% on the second day. The sharp rise is a result of improved personalization of the stress inferencing algorithm. The day 2 classifier was trained using data provided from day 1, and online learning was added to the system, allowing the model to adapt in the field when participants provided stress self-reports. In Study II, three inferences triggered self-reports, walking, speaking, and commuting. 29% of self-reports were triggered by inferences (Speaking: %17, Walking: 9%, Commuting: 3%).

	Requests	Percent Completed	Time to Start	Time to Complete
Study I	566 (26/participant)	80	32 ± 29 (sec)	1.2 ± 0.5 (min)
Study II	629 (17)	87	44 ± 29	2.3 ± 0.8

Table 2: Self-Report statistics.

8.3.3 Behaviors Inferred by mStress

Inferences made by mStress provide some insight into the frequency of behaviors and stress in participants' lives (Table 3. In Study I, 63% of stress inferences were marked as stressed. The implication is that participants were under stress 63% of the time. This seems unlikely and indicates the stress inferencing algorithm may not have performed well when deployed in the natural environment. One reason for this poor performance is the influence of body movement, especially physical activity like walking or running, on the physiological signals. The movement classifier indicated participants had some movement 35% to 37% of the time. In addition, sensor quality degradation likely also played a role.

Stress	Inferences	Inferences / Participant	% Stressed
Study I	14000	350	63
Study II	12600	340	42
Movement			
Study I	20500	911	35
Study II	45000	1241	37

Table 3: Stress and movement inferences made by mStress per day in Study I and Study II.

Table 4 shows the percent of stress inferences that overlap in time with significant movement or sensor degrada-

tion. Movement dominates in both studies, with over 50% of stress inferences likely unreliable due to movement. When combined with unreliability from sensor degradation (ECG and RIP), 58-66% of stress inferences should be considered unreliable and ignored.

	Movement	ECG Error	RIP Error	Overall
Study I	53	6	8	58
Study II	55	15	17	66

Table 4: Percent of stress inferences made during significant movement, degraded ECG signals, degraded RIP signals, and at least one of the above.

Section 5.5 describes an approach to reducing energy consumption that deactivates inferencing when significat movement activity or sensor degradation occurs. Using the statistics in Table 4 and linear approximations to the curves in Figure 6, we can estimate the potential improvement in device lifetime if this energy management technique were implemented. In Study I, we would have a 54% increase from 11 hours to 17 hours in lifetime and in Study II, we would have a 66% increase to 18.5 hours.

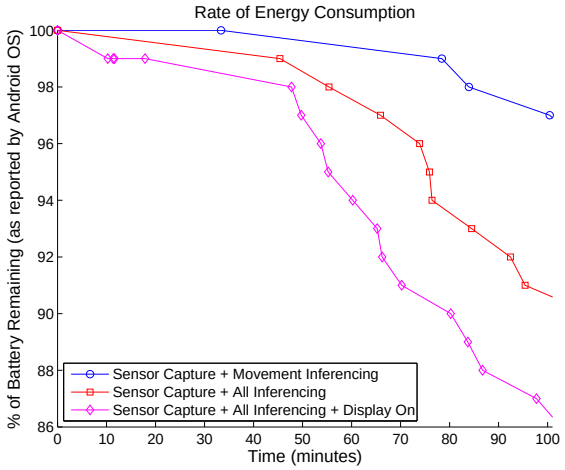

Figure 6: Energy profile of mStress when the system is in it's normal state (data collection + all inferencing), when the participant is completing a self-report (data collection + all inferencing + display on), and when the system is in a reduced power state due to deactivation on significant movement (all inferencing except movement is deactivated). Significant increases in device lifetime occur if the reduced power state is used on significant movement.

9. RELATED WORK

In comparison to existing inferencing systems for mobile phones, mStress is most similar to the MyExperience system. MyExperience [11] collects objective data about study participants on a mobile phone and triggers collection of subjective data from participants based on simple context. The MyExperience architecture is built on 3 core components, sensors, triggers, and actions. Triggers are sets of conditional logic on multi-modal sensor data. When a trigger is true, its corresponding action is taken (e.g., prompt for EMA). mStress shares other characteristics with existing mobile phone context inferencing systems. Two examples include the use of a dynamic activation manager that only enables sensors and features that are needed by active inferencing modules, and reducing resource usage (CPU, battery) by doing a computation once and sharing the result with all modules that need it [19, 33].

The JigSaw framework optimizes sensing pipelines for sensors commonly embedded in smartphones (accelerometer, microphone, and GPS)[22]. JigSaw's accelerometer pipeline produces robust outputs under a variety of positions and orientations. In addition, JigSaw reduces energy consumption by deactivating or throttling down sensing pipeline when behavioral inferences indicate they are unlikely to provide new or high quality data.

mStress differs from the above systems in six ways. First, mStress is capable of more sophisticated inferencing than has been demonstrated in previous mobile-phone-based systems (Section 5.4). It uses sophisticated signal processing of samples to compute features and an SVM to infer if a person is stressed from those features. Sophisticated signal processing is also used in the detection of speaking. Second, mStress uses hierarchical buses to share data and computation among system entities (Section 5). Third, to reduce the delay to produce inferences, feature computation jobs are queued and then processed one-by-one in a single thread (Section 5.3.2). Fourth, mStress is capable of collecting and processing data from both external wireless sensors and sensors embedded in the smartphone (Sections 5.1 and 5.2). Fifth, mStress detects sensor detachments (Section 5.4.4) and provides instructions to the user on how to correct the problem in the field (Section 6.2). Sixth, to our knowledge, mStress is the only mobile sensing framework that has been evaluated in the context of two real scientific field studies, in which over 1,700 hours of sensor data and behavioral inferences were collected from 59 participants.

10. CONCLUSIONS AND FUTURE WORK

To the best of our knowledge, mStress is the first system that produces inferences of stress from physiological measurements in real-time on a mobile phone that is personalized to each participant. It enables robust collection of physiological signals and self-reports close to the occurrence of a stress event. Real-time detection of stress can be used to develop and evaluate personalized coping methods to reduce the adverse impact of ever-increasing stress in people's lives.

Although mStress was successfully used by stress researchers in a one day study in the natural environment of tens of subjects, several improvements are needed to make it suitable and robust for widespread adoption in scientific studies. First, mStress should be more robust to data losses due to lost packets or sensor degradation. mStress can mark these events but has no formal strategy for dealing with them. Ideally, inferencing algorithms would be informed if input data is missing or poor so that a decision can be made as to whether an inference should be made. Alternatively, mStress could switch between multiple implementations of the same inferencing algorithm, each optimized to handle losses of certain features (e.g., one stress inferencing algorithm that uses heart rate, another that uses respiration rate, and another that uses both when both are available). In addition, the Network Layer could impute missing or poor

data from previous measurements. This would make recovery from data losses transparent to the rest of the system.

Second, sensor detachment inferencing should be extended to provide a data quality rating for each window of sensor data. As the ultimate goal of the system is to produce high quality inferences of stress, this rating should be defined in terms of the effect of data quality degradation on the output of stress inferencing. Providing a rating of data quality would allow inferencing algorithms to decide if an inference should be made when presented with a less-than-perfect input window (the norm in natural environments). Similarly, inferences could be tagged with quality scores as well, allowing the mStress framework to decide if an inference should be used to trigger some action (e.g., deactivation of stress inferencing to save energy or triggering of self-reports).

Third, the lifetime of the phone needs to be extended further so it can last more than the awake hours of a day, while supporting network connection to the cloud, active use of other expensive sensors on the phone (e.g., GPS, microphone, etc.) to collect contextual factors, and active usage of the phone for traditional purposes (making calls, browsing, etc.). Systematic approaches for optimizing energy such as event filtering [25, 2] and sampling policy optimization [27, 32] can be investigated. Also, recently developed methods for online learning [26] can be investigated for real-time personalization of stress inferences.

11. REFERENCES

[1] S. Bailey and M. Heitkemper. Morningness-eveningness and early-morning salivary cortisol levels. *Biological psychology*, 32(2-3):181–192, 1991.

[2] A. Bamis and A. Savvides. STFL: a spatio temporal filtering language with applications in assisted living. In *Proceedings of the 2nd International Conference on PErvsive Technologies Related to Assistive Environments*, page 5. ACM, 2009.

[3] L. Bao and S. Intille. Activity recognition from user-annotated acceleration data. *Pervasive Computing*, pages 1–17, 2004.

[4] B. Bruegge and A. H. Dutoit. *Object-Oriented Software Engineering Using UML, Patterns, and Java (3rd Edition)*. Prentice Hall.

[5] E. Butler, F. Wilhelm, and J. Gross. Respiratory sinus arrhythmia, emotion, and emotion regulation during social interaction. *Psychophysiology*, 43(6):612–622, 2006.

[6] G. Clifford. *Signal Processing Methods for Heart Rate Variability*. PhD thesis, Department of Engineering Science, University of Oxford, 2002.

[7] S. Consolvo and et. al. Flowers or a robot army?: encouraging awareness & activity with personal, mobile displays. In *UbiComp*, pages 54–63, 2008.

[8] M. Danninger, T. Kluge, and R. Stiefelhagen. MyConnector: analysis of context cues to predict human availability for communication. In *Proceedings of the 8th international conference on Multimodal interfaces*, page 19. ACM, 2006.

[9] A. Dias and et. al. Measuring physical activity with sensors: a qualitative study. *Studies in health technology and informatics*, 150:475, 2009.

[10] R. Farrell and B. Young. Effect of lead quality on computerized ECG interpretation. *Computers in Cardiology*, 31:173, 2004.

[11] 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 '07: Proceedings of the 5th international conference on Mobile systems, applications and services*, pages 57–70, New York, NY, USA, 2007. ACM.

[12] E. Gamma, R. Helm, R. Johnson, and J. M. Vlissides. *Design Patterns: Elements of Reusable Object-Oriented Software*. Addison-Wesley Professional, 1994.

[13] Google. Designing for Performance | Android Developers, 2009.

[14] J. Healey. *Wearable and Automotive Systems for Affect Recognition from Physiology*. PhD thesis, MIT, 2000.

[15] R. IETF. 1662, PPP in HDLC-like Framing, 1994.

[16] S. S. Intille, E. M. Tapia, J. Rondoni, J. Beaudin, C. Kukla, S. Agarwal, L. Bao, and K. Larson. Tools for studying behavior and technology in natural settings. In *In Proceedings of UBICOMP 2003*, pages 157–174. Springer, 2003.

[17] T. Kamarck, D. Janicki, S. Shiffman, D. Polk, M. Muldoon, L. Liebenauer, and J. Schwartz. Psychosocial demands and ambulatory blood pressure: a field assessment approach. *Psysiology and Behavior*, 77:699–704, 2002.

[18] T. Kamarck, S. Shiffman, L. Smithline, J. Goodie, J. Paty, M. Gnys, and J. Jong. Effects of task strain, social conflict, and emotional activation on ambulatory cardiovascular activity: daily life consequences of recurring stress in a multiethnic adult sample. *Health Psychol*, 17(1):17–29, 1998.

[19] S. Kang and et. al. SeeMon: scalable and energy-efficient context monitoring framework for sensor-rich mobile environments. In *ACM MobiSys*, 2008.

[20] M. Kjaergaard, J. Langdal, T. Godsk, and T. Toftkjaer. Entracked: energy-efficient robust position tracking for mobile devices. In *ACM MobiSys*, 2009.

[21] J. Liu. Subjective Sensing: Intentional Awareness for Personalized Services. In *NSF Workshop on Future Directions in Networked Sensing Systems: Fundamentals and Applications*, 2009.

[22] H. Lu and et al. The Jigsaw Continuous Sensing Engine for Mobile Phone Applications. In *ACM SenSys*, 2010.

[23] U. Maurer, A. Smailagic, D. Siewiorek, and M. Deisher. Activity recognition and monitoring using multiple sensors on different body positions. In *BSN*, page 4, 2006.

[24] D. McFarland. Respiratory markers of conversational interaction. *Journal of Speech, Language, and Hearing Research*, 44(1):128, 2001.

[25] Y. Mei and S. Madden. Zstream: A cost-based query processor for adaptively detecting composite events. In *ACM SIGMOD*, 2009.

[26] E. Miluzzo and et. al. Darwin Phones: the Evolution of Sensing and Inference on Mobile Phones. In *ACM MobiSys*, 2010.

[27] M. Ra and et. al. Energy Delay Tradeoffs in Smartphone Applications. In *ACM MobiSys*, 2010.

[28] N. Ravi, N. Dandekar, P. Mysore, and M. Littman. Activity recognition from accelerometer data. In *National Conference on Artificial Intelligence*, page 1541, 2005.

[29] D. M. Ritchie and K. Thompson. The UNIX time-sharing system. *Communications of the ACM*, 26(1), 1983.

[30] B. Schölkopf and A. Smola. *Learning with kernels: Support vector machines, regularization, optimization, and beyond*. the MIT Press, 2002.

[31] A. A. Stone, S. Shiffman, A. A. Atienz, and L. Nebeling. *The Science of Real-Time Data Capture: Self-Reports in Health Research*. Oxford University Press US, 2007.

[32] Y. Wang, B. Krishnamachari, Q. Zhao, and M. Annavaram. Markov-Optimal Sensing Policy for User State Estimation in Mobile Devices. In *ACM IPSN*, 2010.

[33] Y. Wang, J. Lin, M. Annavaram, Q. Jacobson, J. Hong, B. Krishnamachari, and N. Sadeh. A framework of energy efficient mobile sensing for automatic user state recognition. In *ACM MobiSys*, 2009.

[34] F. Wilhelm and P. Grossman. Emotions beyond the laboratory: Theoretical fundaments, study design, and analytic strategies for advanced ambulatory assessment. *Biological Psychology*, 2010.