Reducing Users' Perceived Mental Effort due to Interruptive Notifications in Multi-Device Mobile Environments

Tadashi Okoshi, Julian Ramos[1], Hiroki Nozaki,
Jin Nakazawa, Anind K. Dey[1] and Hideyuki Tokuda
Keio University, {slash, chacha, jin, hxt}@ht.sfc.keio.ac.jp
Carnegie Mellon University[1], {ingenia, anind}@andrew.cmu.edu

ABSTRACT
In today's ubiquitous computing environment where users carry, manipulate, and interact with an increasing number of networked devices, applications and web services, human attention is the new bottleneck in computing. It is therefore important to minimize a user's mental effort due to notifications, especially in situations where users are mobile and using multiple wearable and mobile devices. To this end, we propose Attelia II, a novel middleware that identifies breakpoints in users' lives while using those devices, and delivers notifications at these moments. Attelia II works in real-time and uses only the mobile and wearable devices that users naturally use and wear, without any modifications to applications, and without any dedicated psycho-physiological sensors. Our in-the-wild evaluation in users' multi-device environment (smart phones and smart watches) with 41 participants for 1 month validated the effectiveness of Attelia. Our new physical activity-based breakpoint detection, in addition to the UI Event-based breakpoint detection, resulted in a 71.8% greater reduction of users' perception of workload, compared with our previous system that used UI events only. Adding this functionality to a smart watch reduced workload perception by 19.4% compared to random timing of notification deliveries. Our multi-device breakpoint detection across smart phones and watches resulted in about 3 times greater reduction in workload perception than our previous system.

Author Keywords
interruption overload; interruptibility; mobile sensing;

ACM Classification Keywords
H.5.m. Information Interfaces and Presentation (e.g., HCI): Miscellaneous

1.INTRODUCTION
There has been an explosion of information available for people to read and act on. However, the amount of attention that can apply to this growing amount of information, has remained constant [28]. Approaches for dealing with this include multitasking or dividing attention among a number of sources, and relying on push notifications to bring information to the forefront of their attention. However, notifications are responsible for an even greater number of interruptions.

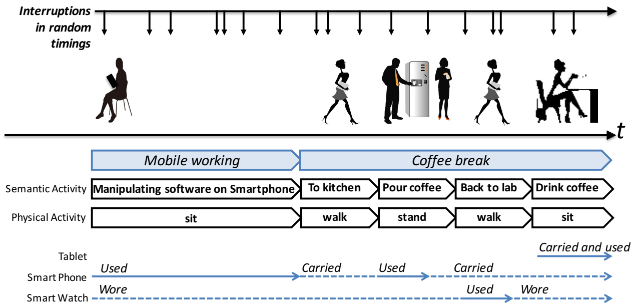

Figure 1. User's Notification Experience Scenario

This is exacerbated by the fact that users are carrying, wearing and using a growing number of mobile and wearable computing devices including notebooks, tablets, smart phones, smart watches or wearable sensors [13, 37], all of which can deliver interruptive notifications. Making the problem even worse are the growing number of installed applications on each device, each of which can also interrupt the mobile device owner. In particular, communication-based applications that support phone calls, texts chats, and social networking particularly suffer from this. But, games, news and other applications have similar issues as well, leading to a setting where people's everyday lives are significantly impacted [6, 33, 42] from a feeling that they are being constantly interrupted by their computing systems.

Given the ever-increasing degree of information overload, the limited resource of human attention is the new bottleneck [9, 38] in interactive computing. In this paper, we particularly focus on **interruption overload**, a form of distraction caused by the excessive number and inappropriate delivery of notifications from computing systems. All widely-used notification systems deliver notifications as soon as they are received, and this has been shown to negatively affect users' work productivity [1, 2, 5, 29]. Imagine the scenario illustrated in Figure 1, where Melissa carries, wears, and uses multiple mobile and wearable devices, including her smart watch on the wrist, a smart phone, and a tablet. In the beginning, she is sitting down and doing work on her smart phone. After a while, she decides to take a coffee break. Melissa stands up, walks to the kitchen, pours coffee, walks back, sits down on the couch and enjoys her coffee, watching video on her tablet. In the current computing environment, Melissa experiences notifications at "random" timings; that is, as they arrive on her devices. In other words, notifications from a variety of applications and services reach Melissa without any consideration of whether she is actually interruptible, causing divided attention and possible having negative impacts on her work productivity. In ubiquitous computing where computer systems promote calmness [44], they need to behave (e.g., delivering notifications in this case) adaptively with regard to the user's current attention and interruptibility status.

To address this problem, we follow the proposal of others (e.g., [24]), to defer notifications until the user is experiencing a natural **breakpoint** [32], defined as the boundary between two adjacent units of a user's activity. Deferring the interruptive notifications to this point can lower the impact of the interruption on users' cognitive load [1, 21, 22]. Our earlier work on **Attelia I** [35] focused particularly on the user's mobile experience when actively interacting with her smart phone, using this information to detect breakpoints. We showed that, by delaying smart phone notifications until a breakpoint is detected, it reduces users' perception of workload due to interruptive notifications by 33%, as measured by using NASA-TLX [16] daily. Attelia I performed this breakpoint detection in real-time, by sensing user interaction (UI) events on users' smart phones and applying machine learning.

In this paper, we extend our earlier work to address the interruption overload problem by leveraging the multiple mobile and wearable devices that users carry in today's world. We demonstrate our ability to detect breakpoints (1) using sensed data from *multiple mobile and wearable devices*, such as a smart phone and smart watch, and (2) during user's daily life with the devices, including *periods in which the devices are carried or worn but are not actively manipulated or interacted with*, in addition to user's mobile interactions. Our new system, **Attelia II**, (1) works on a variety of mobile and wearable devices, not requiring the use of dedicated external psycho-physiological sensors, (2) performs real-time detection of breakpoints to support real-time adaptation, (3) detects breakpoints both while users are actively manipulating their devices and when they are not (e.g., carrying it in a pocket), and (4) is applicable to situations of user mobility and can be used by a wide variety of applications.

Our in-the-wild user study with 41 participants for 1 month proved the value of Attelia II. Introducing physical activity-based breakpoint detection in addition to Attelia I's UI event-based breakpoint detection had a reduction in workload perception of 71.8% greater than that of Attelia I. Porting these detectors to a smart watch reduced user's workload perception by 19.4% compared to delivering notifications at random times. Lastly, our multi-device combinational breakpoint detection model across smart phones and watches reduced users' workload perception by 31.7%, approximately equating to three times greater reduction than with Attelia I.

The contribution of our paper is four-fold.

- We present the first middleware for real-time breakpoint detection leveraging a user's multiple mobile and wearable devices. This novel system does not require the use of dedicated external psycho-physiological sensors nor the modification of any applications on users' devices. To the best of our knowledge, Attelia II is the first system to address user's interruptibility in the mobile multi-device context.
- We show the value of physical activity-based breakpoint detection for reducing users' workload perception.
- We show how our breakpoint detection on the smart watch can effectively reduce workload perception.
- We present the results that our "combinational" breakpoint detection model running on top of multiple mobile and wearable devices resulted in the biggest reduction amongst all models.

INTERRUPTION OVERLOAD

Interruption overload is a sub-component of information overload. It is caused by large numbers of ill-timed notifications, and has only increased in frequency in recent years. Not surprisingly, there has been a greater research focus on the topics of interruptions and multitasking [14]. Rather than forcing users to manually check whether new information is available, notifications instead push new information to users, resulting in faster and increased awareness.

Despite the obvious benefits of notifications, they have profound negative impacts on users' lives. They increase negative affect and emotional states, social attribution [1] and psycho-physiological states [45]. They reduce work productivity and the quality and amount of time available for decision making; they also increase the time that it takes to resume a task after attending to a notification [1, 2, 5, 29, 45, 41]. That is, if they even return to their original task [33]. One common strategy for dealing with these interruptive notifications is to disable them completely or temporarily. However, disabling them negates their benefits, with users not receiving timely information updates. In fact, users prefer to keep using these notification systems for information delivery even given the interruption costs, rather than turning them off and checking for new information manually [25].

Approaches for Mitigating the Cost of Notifications

As opposed to disabling notifications completely, we find that there are two common approaches for addressing interruptive overload in the literature: (a) deferring notifications to a more appropriate time, and (b) mitigating the interruptive nature of notifications. When deferring notifications, an appropriate deferral time must be identified. A number of researchers have identified "breakpoints" in user activities as this deferral time. Breakpoints [32] are a concept in psychology in which a human's perceptual system segments activities into a hierarchical structure of discrete sub-actions. The boundary between two adjacent action units is called a breakpoint. Deferring notifications until a detected breakpoint has been shown to reduce interruption cost in terms of task resumption lag and subjective frustration [1, 21, 22]. The other approach, mitigation, tries to reduce a the impact of the notification on a user's workload perception by changing the modality used to deliver notifications. This can include the use of "silent" mode, "vibration" mode or only flashing an LED (e.g., [31]). This approach serves to change the saliency of the interruption, while leaving the timing of the notifications unchanged.

While these two approaches are complementary, we focus on notification deferral. Given the growing number of notifications, changing the timing of notifications rather than their saliency would seem to have greater potential impact on users' interruption overload. With this focus, we turn our attention to identifying interruptive moments or breakpoints.

IDENTIFYING INTERRUPTIBLE MOMENTS

Early work on identifying breakpoints naturally focused on desktop environments. For example, Horvitz et al. inferred interruptibility accurately in desktop computing environments, by using context information, such as interaction with computing devices, visual and acoustical analyses, and online calendars [18]. Hudson et al. constructed statistical models for predicting office workers' interruptibility, using long-term audio/video recordings with manually-emulated sensors of user's activity status, along with experience sampling technique [20]. For these systems, recognition was performed in an a posteriori manner. Later work by Begole et al. [3] and by Horvitz et al. [19] focused on systems that supported real-time detection of interruptibility, but these systems required the use of dedicated custom hardware. In contrast, OASIS also identified breakpoints in real-time, but did no require custom hardware, instead using information about user interactions with an application and user-provided annotations [24]. OASIS deferred the deliver of desktop-based notifications until a breakpoint was detected. While both our system and OASIS use breakpoints, there are some significant differences. OASIS only focused on users interacting with devices in desktop computing, while our solution focuses users interacting with multiple devices while mobile, including both user-device interaction and physical activities. While OASIS employed post-hoc breakpoint annotation, our system uses a real-time annotation scheme. Finally OASIS was evaluated in the lab with a specific set of applications, while our valuation was performed in-the-wild, on user's own devices with their own applications.

Detecting Interruptibility in Mobile Environments

Breakpoint detection research has also been conducted in the context of mobile devices. For example, Ho et al. used wireless on-body accelerometers to trigger interruptions when users transitioned between activities [17]. Interruptions delivered at these transition times reduced user annoyance. This approach is promising, but required the use of an external on-body sensor. Fischer et al. also identified breakpoints based on transitions between activities, but focused on moments immediately after phone-based activities including the completion of phone calls and text messages [8]. Users tended to be more responsive to notifications after these activities than at random other timings. Again, this approach is promising, but is limited to a small set of communications activities.

Other researchers have focused on using a wider variety of user context to determine moments of interruptibility. For example, Hofte et al. used an experience sampling methodology to collect information on location, transit status, company and activities in order to build a model of interruptibility [43] especially for phone calls. Pejovic et al. expanded the use of context for detecting moments of interruptibility on smartphones including user activity, location, time of day, emotions and engagement. Their system, InterruptMe, uses this information to decide when to interrupt users [36]. While their system needs manually-provided information about the user's interruptibility, such as company or emotion, our system simply relies on the sensor data from user's devices and does not need any manual input.

Other work in mobile computing has focused on mitigating the impact of notifications. This is a complementary approach to our focus on deferring notifications. Smith et al. attempted to mitigate the impact of disruptive phone calls, by automatically setting phone call ring tones to different modes, such as silent answering, declining, and ignoring [39]. Their user study showed that this approach of identifying which ring tone to use was useful, even when underlying user behavior changed. Böhmer et al. also focused on incoming phone calls and explored the design space of incoming call alerts to users on smart phones, instead of conventional full-screen notifications [4]. Their proposed strategies include "postponing" the call acceptance and "multiplexing" the alert screen. The multiplexing approach obtained the best evaluation in their large-scale user study. Their "multiplexing" approach can also be combined with our deferral approach. The "postponing" approach looks similar to our deferred scheduling approach, but is specifically focused on phone call notifications.

Attention with Multiple Devices

There is also research on attention-awareness in multi-device environments. Dostal proposed DiffDisplays [7], a system for tracking the display the user is currently looking at by cameras and computer vision. Inspired by several techniques for visualizing changes in unattended displays, Garrido proposed AwToolkit [10] for developers to support maintaining user's awareness in multi-display systems. The toolkit detects which display is currently being looked at by the user and provides interruptive notifications with multiple different levels of "subtlety" to draw the user's attention to unattended changes in the displays. Although this Gaze-tracking technique can be adopted in mobile environments, it is not considered to be compatible with diverse low-computation mobile and wearable devices, such as watches or bands, especially for the purpose of attention target classification. Attelia currently uses display on/off event for such classification. On the other hand, for activity recognition, gaze-tracking or blink-tracking has been used [26], thus it has potential as a source of information to use for breakpoint detection.

PRINCIPLES FOR DETECTING BREAKPOINTS

As just presented, there has been a lot of research conducted in identifying breakpoints for the purpose of deferring notifications. Our own research on detecting breakpoints builds on this work in the following ways:

- **Is feasible for mobile and wearable devices:** Users carry and use multiple mobile and wearable devices for everyday computing. A breakpoint detection system needs to be able to run on such platforms. It should not require the use of any external hardware, other than what users already carry.
- **Supports real-time detection:** To support notification adaptation and deferral on the fly, the breakpoint detection needs to be performed in real-time.
- **Can be applied to diverse types of notification sources:** The breakpoint detection system needs to work for diverse types of notification sources, and not for only e.g., phone calls and text messages.
- **All-day-long use:** Breakpoint detection needs to be performed all day long, no matter what the user is doing.

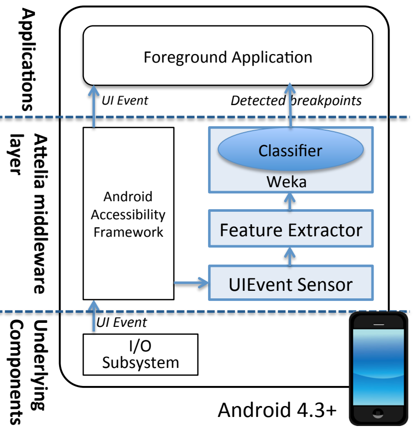

Figure 2. Attelia I System Architecture

ATTELIA I

With **Attelia I** [35], our first system, we *partially addressed* the features described above for breakpoint detection. Attelia I focused on smart phones only and only detected breakpoints while the user is *actively manipulating* her phone. Attelia I is the first system that detects user's breakpoint (1) solely on smart phone without any dedicated psycho-physiological sensors, (2) in real-time, (3) and works with a wide variety of smart phone applications.

Our approach in Attelia I was to collect the UI Event stream generated when a user manipulates and interacts with applications on the smart phone on the go. This event stream was used as sensor data and fed to a J48 classifier running on the smart phone to detect breakpoints. Figure 2 shows the system architecture of Attelia I. As a middleware service using Android's "Accessibility Framework", Attelia reads the event stream from the active application currently being manipulated by the user, and periodically classifies whether the current time frame is a breakpoint by using an instance of the Weka machine learning engine.

A controlled in-lab user study showed that delivering notifications at detected breakpoint timings resulted in 46.2% lower workload perception compared to randomly-timed notifications which emulate the "current" notification experience before Attelia. Further, an in-the-wild user study with 30 participants for 16 days validated Attelia's value, with a 33.3% decrease in workload perception compared to randomly-timed notifications.

DESIGN OF ATTELIA II

Encouraged by our promising results with Attelia I, in this paper, we build on our past work in two novel and important ways. First, we address how breakpoint detection can be applied in the **multi-device** (i.e., smart phones and smart watches) ubiquitous computing environments that users are often in, and use these devices to detect breakpoints. Second, Attelia I only detected breakpoints during active interaction with a smart phone. In Attelia II, we extend the breakpoint detection to cover all aspects of a user's daily life, including the period the devices are carried or worn but not actively manipulated. We demonstrate the impact of this increased coverage on users' workload perception.

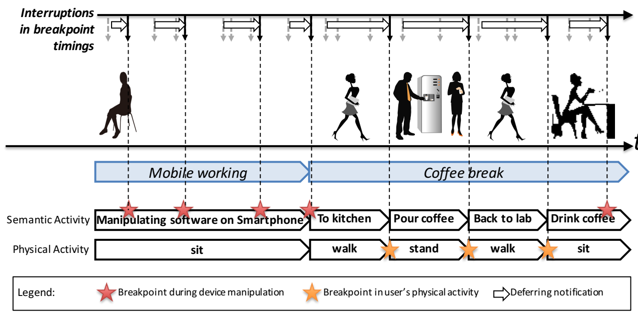

Figure 3. User's Notification Experience Scenario with Attelia II

Figure 3 revisits the same scenario of Melissa shown in the Introduction section and shows how Attelia II helps Melissa in this situation. With multiple types of sensing techniques, Attelia II detects her breakpoints, both during her active device manipulation (interacting with applications on her smart phone) on the go and during her physical activity. Notifications from a variety of applications and services, originally delivered to Melissa at random timings without Attelia, are now delivered to her at the detected breakpoint timings. This notification delivery is less interruptive and lowers Melissa's workload perception.

We now describe how Attelia II addresses the four principles for detecting breakpoints, presented earlier. In the following subsections, we provide details of how Attelia II satisfies these four features.

1. **Detection on mobile and wearable devices**: Attelia II detects breakpoints on a variety of mobile and wearable devices, such as the smart phone, smart watch and the tablet in Melissa's scenario in Figure 3, without the use of an external server or any psycho-physiological sensors, such as an ECG sensor.
2. **Real-time detection**: Attelia II detects breakpoints in real-time (not post-hoc), thus it can be used to adapt notification timings at run-time.
3. **High applicability for a variety of applications:** Attelia II leverages user interface events *and* physical activity events, hence it can work with any application installed and running on users devices. No modification of installed applications is needed for Attelia to function.
4. **Opportunistic breakpoint detection in users' everyday life using their devices**: Attelia II detects opportune times for notifications not only during a user's active interaction (manipulation) with her devices, but also in other non-active periods, such as when she carries the mobile wearable devices but is not actively manipulating them. In Melissa's scenario, breakpoints during application usage on her smart phone will be detected, as will breakpoints during her physical activities (e.g., walking, standing, or sitting) that are detected in the second phase. With this hybrid approach, Attelia II leverages information from her multiple devices to detect breakpoints throughout her day.

"Breakpoint" as a Temporal Target for Interruption

Referring to the results from our previous work and other related research, Attelia II uses "breakpoints" [32] as a temporal target for sensing an "opportune moment" for delivering interruptive notifications with reduced user workload perception. Related work in real-time sensing of available user attention or cognitive load using psycho-physiological sensors showed that at least two psycho-physiological sensors are needed even in non-mobile situations [15]. Given the burden of wearing a psycho-physiological device constantly, our approach only uses the users' mobile devices, and attempts to sense more coarse-grained, but easier to sense signals, from which appropriate timings for notifications can be inferred. In Attelia II, we introduce two different notions of breakpoints, namely **User Interaction-based Breakpoint** and **Physical Activity-based Breakpoint**.

User Interaction-based Breakpoint

While the user is manipulating a device that he is carrying or wearing, there is an application that is the target of his manipulation. Thus, for the period when the device is actively being manipulated, we focus on the interaction between the user and the application and use that information for detecting a user's breakpoints. Although the application itself is one possible source of knowledge about breakpoints, using knowledge from the internals of any specific application is not feasible nor scalable, given the huge number of applications available and the fact that application developers would need to expose internal information at development time. Instead, we collect run-time status events from the operating system and executing applications, and use them to identify relationships to ground truth values of interruptive overload provided by users, during a training phase. During this training phase, users indicate when they are interruptible by pressing an always-present button on their interface. This training data is provided to a J48 classifier running on the mobile device. Note that the UI-based Breakpoint Detection described here is the same as what was presented in Attelia I. For more details, refer to [35].

Physical Activity-based Breakpoint

In our daily lives with smart phones and smart watches, there is a significant amount of time when we just carry or wear them but do not actively use (manipulate) them. For example in Melissa's scenario, she wears her smart watch and carries her smart phone in her pocket but does not actively manipulate them while moving from the lab to the kitchen, getting coffee and returning to the lab. Another example is when a user is just reading a book, sitting on a sofa, and wearing his smart watch. To comprehensively address interruption overload in a user's daily life, we need to handle this type of situation, by finding an opportune moment to deliver notifications while users are not actively manipulating their devices.

To this end, we focus on transitions in a user's physical activity, such as "when a user stands up" or "when a user stops running". We specifically hypothesize that when a person changes her activity from a high energy state to a lower energy state, that timing can be strongly considered as her breakpoint. (Later we validate this hypothesis with input gathered from users.) Concretely on the mobile and wearable devices, Attelia II declares a physical activity-based breakpoint when such a change in the user's activity is detected, using activity recognition mechanisms built on top of the hardware sensors already available on mobile platforms, such as the accelerometer or GPS.

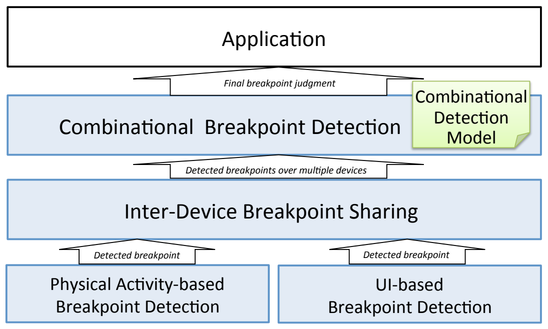
Application
Final breakpoint judgment
Combinational Breakpoint Detection
Combinational Detection Model
Detected breakpoints over multiple devices
Inter-Device Breakpoint Sharing
Detected breakpoint
Detected breakpoint
Physical Activity-based Breakpoint Detection
UI-based Breakpoint Detection

Figure 4. Attelia II Layered Breakpoint Detection Model

Mobile Sensing to Real-Time Breakpoint Detection

In order to realize real-time detection on multiple mobile and wearable devices, Attelia II has its own model for overall breakpoint detection shown in Figure 4. (1) On each device, both the "UI-based Breakpoint Detection" (while the device is being actively manipulated) and "Physical Activity-based Breakpoint Detection" (while the device is not being manipulated) will be running, according to the current device usage. (2) Each detection component executes its own local binary classifier to detect breakpoints at a configured periodicity and outputs the binary value. Those local classification outputs along with the device usage status information will be exchanged across a user's multiple devices via an "Inter-Device Breakpoint Sharing" layer. (3) A "Combinational Breakpoint Detection" algorithm reads the current values of all underlying local breakpoint detectors and device usage statuses, and generates a **final decision on the user's breakpoint status** across devices, based on the selected "Combinational Detection Model".

ATTELIA II SYSTEM ARCHITECTURE

Based on the design in the previous section, we describe the system architecture of Attelia II implemented on the Android platform in this section. Figure 5 shows the system structure of Attelia II prototype implemented on the generic Android 4.3 (and above) platform and Android Wear 5 (and above) platform. The current prototype runs on a variety of Android devices including smart phones, tablets, notebooks, smart cameras, and smart watches. Attelia II is implemented as a middleware service for the Android platform and runs on each device of the user. The middleware implementation allows the service to be distributed through the Google Play store and contributes to the deployability of the system to end users.

Attelia II uses several underlying mechanisms inside the Android platform (along with additional components we implemented). Each individual breakpoint detector reads a data stream from the underlying systems, such as activity recognition results or the UI event stream and detects breakpoints by using its own feature extraction and classification (powered by Weka [30]) logic, respectively. These detection results are exchanged among multiple devices over Bluetooth-based PAN. If no breakpoints are detected by the low-level detectors, nothing is done. However, when a breakpoint is detected, the Combinational Breakpoint Detector combines these results, according to a configured Detection Model, and produces a final breakpoint judgment.

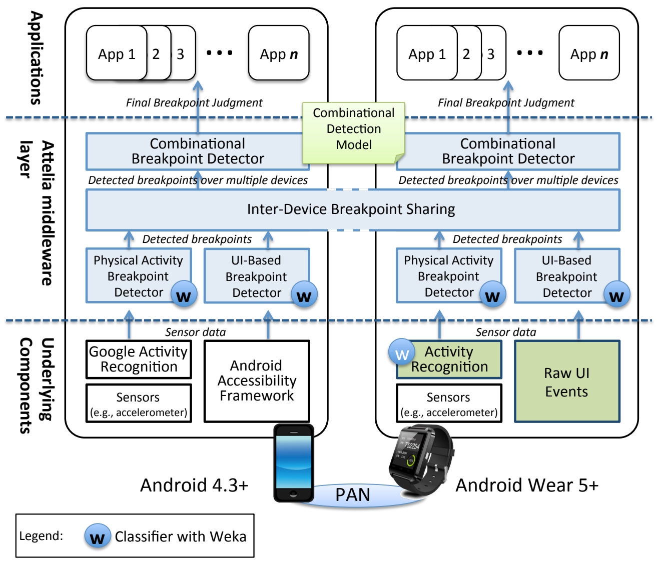
Applications
Attelia middleware layer
Underlying Components
App 1
2
3
App n
Final Breakpoint Judgment
Combinational Detection Model
Combinational Breakpoint Detector
Detected breakpoints over multiple devices
Inter-Device Breakpoint Sharing
Detected breakpoints
Physical Activity Breakpoint Detector
UI-Based Breakpoint Detector
Sensor data
Google Activity Recognition
Sensors (e.g., accelerometer)
Android Accessibility Framework
Activity Recognition
Raw UI Events
Android 4.3+
PAN
Android Wear 5+
Legend:
Classifier with Weka

Figure 5. Attelia II System Architecture

Each device runs an identically selected Combinational Breakpoint Detector, that has access to the same information as the others (e.g., which detectors detected a breakpoint and which devices are actively being used). When they make a final judgment that a breakpoint has occurred, they use the device usage information to determine which device should deliver any deferred notifications. For example, if the phone is being used, the phone should receive the notifications since it already has the user's attention.

UI-based Breakpoint Detection

Table 1 shows the list of mechanisms used for each detector. On the generic Android platform, Attelia II uses the same UI-based breakpoint detection mechanism as Attelia I. Using the Android Accessibility Framework [11], the detector reads the UI event stream, extracts feature vectors, and executes a J48 classifier on Weka every 2.5 seconds, while the device is being actively used (more specifically, while the device's screen is on and there is any UI event fired in this time frame).

On the Android Wear 5 platform, since the Android Accessibility Framework is not provided as of version 5.0.1, we implemented a breakpoint detector which uses the Linux Input Subsystem. Due to the nature of available Android smart watches and the fact that most current "Android Wear" products mainly support checking (Android's) notifications, we take any manipulation on the watch screen as an indication that the user is at a breakpoint. Thus, the current breakpoint detector implementation looks for breakpoints every 2.5 seconds, if more than one tap-related event comes from the underlying Linux Input Subsystem in this time window.

Physical Activity-based Breakpoint Detection

Physical activity-based breakpoint detection is based on **a transition** in a user's physical activity, such as when she stops walking. This breakpoint detector relies on underlying activity recognition that generates labels such as walking, running, or still, and detects breakpoints according to their changes.

Breakpoint Detector	Generic Android	Android Wear
UI-based Detector	Accessibility Framework [11]	Linux Input Subsystem
Physical Activity -based Detector	Google Play Services Location APIs [12]	Original accelerometer -based activity recognition

Table 1. Breakpoint Detection Mechanisms in Attelia II

		To				
		onbike	running	walking	working	still
From	onbike		4.7 (3.1)	**6.8** (2.5)	4.9 (3.4)	**6.4** (3.0)
	running	4.7 (3.0)		**8.2** (1.4)	4.5 (3.3)	**7.0** (2.6)
	walking	4.3 (3.0)	5.0 (2.9)		**5.3** (3.3)	**7.4** (2.3)
	working	4.8 (3.5)	**5.4** (3.1)	**6.9** (2.6)		**5.8** (3.6)
	still	4.7 (3.3)	**5.1** (3.1)	**7.3** (2.3)	3.8 (2.9)	

Table 2. Ground Truth on Physical Activity Change Breakpoint
Each number shows the average (and standard deviation) using a 10-point Likert scale. Values in bold indicate those used by the breakpoint detector.

		Classified As		
		still	walking	running
Ground truth	still	92.2%	7.7%	0.0 %
	walking	4.3%	95.1%	0.6 %
	running	5.1%	5.3%	89.8%

Table 3. Confusion Matrix: Cross Validation of Activity Recognition

To confirm our earlier hypothesis about detecting breakpoints in during changes in physical activities (i.e., that breakpoints exist when moving from a high-energy to a low-energy activity), we conducted a survey. The survey asked participants to rate, using a 10-point Likert scale, the likelihood of a breakpoint when transitioning between each pair of the following physical activities: bike-ride, running, walking, working at a desk, and being still. Table 2 summarizes the results from 26 university students. Based on these results, we built a model for detecting physical activity-based breakpoints, using values greater than 5 as indicators of breakpoints.

Generic Android Platform

On the generic Android platform, physical activity-based breakpoint detection is built on top of the "ActivityRecognition" API of Google Play Service Location APIs [12]. Using sensors and the GPS embedded in the Android devices, this API returns the device's current activity, such as "STILL", "IN_VEHICLE", "RUNNING", or "ON_BICYCLE". The frequency with which the API returns the activity depends on the Android platform version. Using this activity information, our current Attelia II implementation classifies the activity changes as breakpoints according to Table 2.

Android Wear Platform

On the Android Wear 5 platform, we implemented our own activity recognition since the API described above is not supported. We built an accelerometer-based activity recognizer according to [34]. Our implementation uses the Sony SmartWatch3 [40], which collects sensor data at 50Hz. Based on previous literature on activity recognition [34], we use 22 commonly used time-domain and frequency features extracted over a 3-second sliding window and a J48 decision tree classifier implemented with Weka [30]. Due to the reduced capability of Android Wear devices, our current implementation only classifies "still", "walking", and "running". We trained a J48 model with ground truth collected from 10 users. The cross validation result is shown in Table 3.

Inter-Device Communication

Attelia II shares the breakpoint detection events and device-usage status across multiple devices via Bluetooth. When a local breakpoint detector detects a breakpoint, attributes about the breakpoint, including its timestamp and detector type, will be sent to other devices in real-time. This assumes that the user is wearing and carrying devices that are within the range of Bluetooth wireless communication. For device-usage status, Attelia II sends "DEVICE IN_USE" to remote devices when the screen of the local device turns on, and "DEVICE NOT_USED" when the screen turns off. This simple implementation covers most situations in which a user is manipulating target devices, since most of the time the screen is on when the user is interacting with the a device.

Combining Breakpoint Detection

Every time a breakpoint detection event comes from any User Interaction-based or Physical Activity-based Breakpoint Detector, our Combinational Breakpoint Detector makes the final decision about whether there is currently a breakpoint, by considering the output of all detectors. Any breakpoint detected by an individual detector within the last 10 seconds is considered to be a "current" breakpoint.

EVALUATION

Using the Attelia II prototype described in the previous section, we conducted an in-the-wild user study with 41 participants for 1 month to evaluate how Attelia II performs in users' multi-device environments. Our objectives in this study were as follows:

1. The Attelia I system used only User Interaction-based breakpoint detection on smart phones. We investigated whether the addition of Physical Activity-based breakpoint detection on smart phones would result in reduced workload perception when dealing with notifications.
2. We wanted to understand the value of having breakpoint detection on a worn smart device. So, on the smart watch alone, we compared the impact of performing breakpoint detection for delivering notifications compared to random delivery timings.
3. As there are different possible ways to combine the different detectors for making a final decision about whether a breakpoint has occurred, we compared different combinations of watch and phone breakpoint detectors to each other, to random delivery and to Attelia I.

Participants

41 (31 male and 10 female) participants were recruited for the study. The participants were university students and staff members, with ages between 19 and 26. 24 participants came from computer science and information technology related departments, while the other 17 came from other schools, such as economics, psychology, or social sciences. All of the participants were smart phone (Android OS version 4.3 or above) users in their daily lives. None had a smart watch, thus we provided each with a Sony SmartWatch3 device to use during the study. Subjects were paid $100 for their participation, and were eligible to win 1 of 2 smart watches via a lottery.

Model Name	Detectors used for combinational breakpoint detection			
	Watch		Phone	
	UI-based	Activity-based	UI-based	Activity-based
Random	None of detectors used. In random timings.			
Phone_UI	Not used	Not used	Used	Not used
Phone_UI_Act	Not used	Not used	Used	Used
Watch_UI_Act	Used	Used	Not used	Not used
Combo(c)	Used	Not used	Used	Not used
Combo(d)	Used	Not used	Not used	Used
Combo(e)	Not used	Used	Used	Not used
Combo(f)	Not used	Used	Not used	Used
Combo(g)	Not used	Used	Used	Used
Combo(h)	Used	Not used	Used	Used
Combo(i)	Used	Used	Not used	Used
Combo(j)	Used	Used	Used	Not used
Combo(k)	Used	Used	Used	Used
Combo(x)	OR(Combo(h), (g), (f), (j), (d))			

Table 4. Combination Breakpoint Detection Models

Overview of the Experiment Procedure

Our experimental procedure consisted of three parts. (1) At the beginning of the study, each participant received instructions for the study, signed our consent form for participation, and received a Sony SmartWatch3. We paired the watch to the participant's phone and installed Attelia II on both devices.

(2) Starting from the next morning, the data collection and breakpoint detection began and lasted for 31 days. Every day, each user experienced Attelia's interruptive notifications, whose timings are based on a randomly selected "breakpoint detection model" from those models shown in Table 4. Attelia II contained definitions of all these models, but these were hidden inside Attelia, thus users did not know which model they were being exposed to each day. Everyday, both devices were set to the same chosen model.

To explore our third objective, we split the 31-day experiment into 2 phases. Table 5 shows the number of days that each model was configured to be selected during each phase. During the first phase, a special "comparison" mode, described later, was configured to collect data efficiently on multiple different combinations of models. During the 2nd phase, the model was changed randomly, but evenly, among the specified 5 models everyday, to prevent ordering effect.

(3) After 31 days, participants filled out the post-experiment survey, uninstalled the Attelia service, returned the watch (except for the lottery winners), and were paid.

Experimental Setup

Combinational Breakpoint Detection Model

In order to achieve all the objectives described above, we created a series of combinational breakpoint detection models as shown in Table 4. Each strategy has a different set of underlying detectors to be used for the combinational detection.

The **Random** model does not use any detectors and displays notifications using random timings. This model emulates what people are currently interrupted by notifications.

Phone_UI and **Phone_UI_Act** are prepared for the first objective. Phone_UI is actually the Attelia I system, which

Phase	Phase 1 (14days)	Phase 2 (17 days)				
Model	*(special "comparison" mode)*	Random	Phone_UI	Phone_UI_Act	Watch_UI_Act	Combo(x)
Duration (days)	14	3	3	3	3	5

Table 5. Phase, Used Model and Duration during the 31 Day User Study

uses only a UI-based detector on the phone. This model delivers notifications at the breakpoint timings detected by the UI-based detector while the device is manipulated (the screen is on), and shows notifications in the random timing while the device is not used (the screen is off). On the other hand, Phone_UI_Act adds the use of the physical activity-based detector. The difference between the two models is that Phone_UI_Act delivers notifications at the breakpoint timings detected by activity-based detector while the device is not used, instead of the random timings.

Watch_UI_Act, along with **Random**, describes the conditions for the second objective. Watch_UI_Act delivers notifications at the breakpoint timings detected by the UI-based detector while the watch is manipulated (the screen is on), and delivers notifications at the breakpoint timings detected by the watch's activity-based detector while the watch is not used (the screen is off).

Combo(c) through **Combo(x)** are the models which involve multiple detectors across devices and were compared for the third objective. These "Combo" models internally use "AND" logic over multiple underlying detectors to make their final breakpoint decision. We used these models to explore whether multiple agreements amongst individual breakpoint detectors might perform better than the individual ones.

Interruptive Notifications

The interruptive notification took the form of a full-screen Experience Sampling Method (ESM) question that asked users to indicate whether the current moment was a good time to be interrupted, using a 5-point Likert scale (1=strongly disagree to 5=strongly agree). This custom notification was employed due to the limitation on Android OS where third party software cannot control timings of Android's "official" notification system. If the user was actively manipulating the smart phone, then the notification was delivered on the phone. Otherwise, we defaulted to delivery on the watch. All the notifications were treated equally without concept of "importance". The minimum interval between two consecutive notifications was set to 1500 seconds, the maximum interval to 1800 seconds, and the daily maximum number of notifications to 20. The study software also was configured to only send notifications between 8AM to 8PM daily. The parameter values were carefully chosen after interviewing prospective participants about their daily lives, to acquire a sufficient number of data samples without overburdening them.

Measurement

We collected user's answers to ESM notifications as described above. In addition, each night, users were given the NASA-TLX [16] survey (on the web), a validated instrument for assessing user's workload. The users were asked to review their notification experience of the day with the selected notification delivery strategy by Attelia, and to evaluate their perception of the workload subjectively.

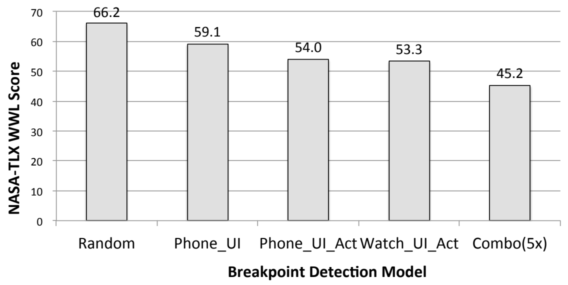

Figure 6. NASA-TLX WWL Scores

Collected Data

Analyzing all the collected data uploaded to our server during 31 days, the average daily duration of device operation, during the times when each of two UI-based breakpoint detectors were active, was 227 minutes on the phone and 1.4 minutes on the watch. When we group operations that are separated by less than 60 seconds, the per-user daily average number of device operations are 174 on the phone, and 9.5 on the watch. Also, the average number of displayed notifications for each user was 10.5 times per day, with 7.3 notifications of these being attended to by the user.

Result: Value of Physical Activity-based Breakpoint Detection

Our first experiment was to investigate whether the addition of physical activity-based breakpoint detection to the already existing UI-based detection on smart phones, would reduced user's workload perception when dealing with notifications. We evaluated these two approaches (Phone_UI and Phone_UI_Act) and "Random" for each user, over a period of 9 days, and compared the resulting workloads.

Figure 6 shows the average TLX Weighted Workload (WWL) scores among the models. The Phone_UI_Act model results in significantly lower workload perception, compared to the Phone_UI (Attelia I) model and Random, which approximates how people are currently interrupted by notifications. When compared to the baseline (Random), Phone_UI_Act had a lower score by 12.2 (i.e., reduced workload), while Phone_UI reduced the workload score only by 7.1. The relative gain (or reduction in workload perception) from using the Phone_UI_Act model is 71.8%, compared to the Phone_UI (Attelia I) model.

A Friedman test revealed a significant effect of notification strategy on the WWL score ($\chi^2(4) = 18.5, p < 0.01$). A post-hoc pair-wise comparison using Wilcoxon rank sum tests showed the significant differences between Random and Phone_UI ($p < 0.05$, $\gamma = 0.28$), between Random and Phone_UI_Act ($p < 0.05$, $\gamma = 0.50$), and between Phone_UI and Phone_UI_Act ($p = 0.05$, $\gamma = 0.24$). Therefore, we can confirm our first hypothesis that adding physical activity breakpoint detection to smartphones is an improvement over just having UI-based breakpoint detection.

Model	(c)	(d)	(e)	(f)	(g)	(h)	(i)	(j)	(k)	Random
(1)Average gain across highest-gainers	1.28	**1.46**	1.17	**1.45**	**1.62**	**1.72**	1.40	**1.47**	(N/A)	(1.00)
(2)Number of displayed ESM	300	46	194	170	68	8	6	72	0	1043
(3)Number of answered ESM	249	34	165	137	54	7	6	58	0	647

Table 6. ESM Score Results on "Combo" Models

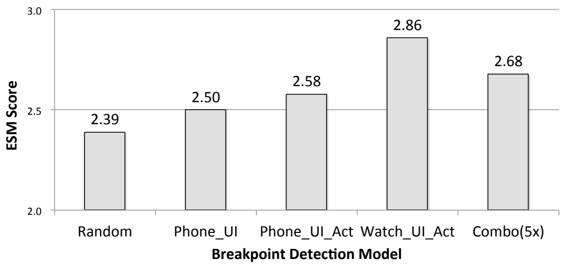

Figure 7. ESM Scores

Figure 7 shows the average ESM interruptibility (5-point Likert scale) scores among the models. The Phone_UI_Act model was rated as providing more appropriate interruptions, when compared to both Phone_UI (Attelia I) and Random.

Result: Attelia II on the Smart Watch

Our next experiment was to investigate whether Attelia II on the watch would reduce user's workload perception. Taking a similar approach as for the previous experiment, we evaluated the Watch_UI_Act model for each user, over a period of 3 days and compared the resulting workload to that of the Random model from the previous experiment.

In Figure 6 the Watch_UI_Act model results in significantly lower workload perception, compared to the Random model (a reduction in workload score of 12.8, or 19.4%). A pairwise comparison using Wilcoxon rank sum tests showed the significant differences between Random and Watch_UI_Act ($p < 0.05, \gamma = 0.35$).

Result: Inter-Device Combinational Models

The final experiment was to investigate the power of inter-device combinational breakpoint detection. Since the number of combinational models ("Combo" models in Table 4) are large, we split the experiment into two phases.

Phase 1: Choosing the Best Model

For the first phase of the study, which lasted for 14 days, our goal was to evaluate the accuracy of the different combinational models: Combo(c) through Combo(k). To do so, we configured Attelia into a special "comparison" mode. We set all four breakpoint detectors to be active (phone UI, phone physical activity, watch UI, watch physical activity). Whenever any of the detectors detected a breakpoint, an ESM-based notification was delivered to the user. The notification asked users to indicate whether it was delivered at an interruptible moment, using a 5 point Likert-scale. In addition, we examined the state of the other three breakpoint detectors. With the state of all four breakpoint detectors, and the ESM value, we could assess the value of all 9 combinational models. For example, consider a situation when a user is interacting with her smart phone and the smart phone's UI-based breakpoint detector is triggered. In addition, her watch's UI-based breakpoint detector was not triggered because she was not manipulating it, but both the watch's and the phone's physical activity-based detectors were triggered because she transitioned from walking to being still. This combination of detectors corresponds to the Combo(g) model. Using the gathered ESM response, we can assess whether this combination accurately detected a breakpoint.

Note that we capped the daily number of breakpoints to 20. Our goal was to acquire 2 ESM responses for each of the 9 combinational models and the Random model each day. To ensure that we achieved this goal, if the Combinational Detector finds a model that can be evaluated at this moment (e.g., Combo(h)), and if the model has not already had its two ESM responses, only then will the ESM notification will be displayed. Otherwise, no notification is delivered. To collect 2 responses for the Random model, "Random"-based notifications were randomly triggered twice daily.

Table 6 summarizes the results. For each user and for each Combo model, we calculated the average ESM score (5-point Likert scale) and compared it to the average ESM score for the Random model. We define the difference between these two scores as the "gain" value. We then looked at which model provided the biggest gain for each user. Row (1) shows, for each model (Combo(c) through (j)), the gain averaged across the users for whom that model had the highest gain. We can clearly see that, for the models that use more of the underlying detectors, (i.e., (g), (h), (i), and (j)), the gain is higher. Note that Combo(k) notification, which reflects the situation where all of 4 detectors detecting breakpoints, did not occur during our study. However, as Row (2) shows, we also observed that the number of answered ESMs (and correspondingly, the number of delivered ESMs - Row 2 - is quite small for those models. This is expected as it is less likely that 3 or more detectors will be active at any given time.

With these results, we chose the best 5 models (those with the highest average gain), and combined them with a disjunction (i.e., a logical OR). We label this new model as **Combo(x)** and we use this in phase 2 of our user study where we compare it to the non-combinational models described earlier.

Phase 2: The Power of the Best Model

After the first phase ended and we successfully created the model Combo(x), we began phase 2 which lasted for 17 days. We evaluated this model with our other models: Random, Phone_UI, Phone_UI_Act, and Watch_UI_Act. Note that experiments 1 and 2 were conducted during phase 2, and it is those results that we compare to Combo(x). All users experienced all models for at least 3 days, in a random ordering.

Again, Figure 6 shows the comparisons of average TLX WWL scores across the models. The Combo(x) model has a significantly lower workload than the other models, including Phone_UI_Act and Watch_UI_Act. Combo(x) amazingly resulted in an 295.1% more reduction in workload perception, compared to Phone_UI (Attelia I). When compared to the baseline (Random), Combo(x) resulted in a score that was 21.0 points lower, while Phone_UI resulted only in a reduction of 7.1. A post-hoc pair-wise comparison using Wilcoxon rank sum tests showed the significant differences between "Random" and "Combo(x)" ($p < 0.01$, $\gamma = 0.71$), between "Phone_UI" and "Combo(x)" ($p < 0.01$, $\gamma = 0.51$), and between "Phone_UI_Act" and "Combo(x)" ($p = 0.05$, $\gamma = 0.25$).

DISCUSSION

In our in-the-wild user study, we demonstrated the value of Attelia II in a multi-device environment. We showed that an introduction of physical activity-based breakpoint detection in addition to Attelia I's UI event-based breakpoint detection resulted in a 71.8% greater reduction in users' workload perception. Using both of these detectors on the smart watch resulted in a reduction of 19.4% in user cognitive load, compared to random notification delivery. Finally, our best multi-device combinational breakpoint detection model "Combo(x)" resulted in a 31.7% lower workload perception compared to the random case, which is almost three times greater reduction than that in Attelia I. Now we discuss further research opportunities and challenges that this result enables.

Figure 7 shows the results of the average ESM scores for each breakpoint detection model. The average scores across the models show promising values in terms of users' self-reported interruptibility. However, there were no statistical significance between the different models. We hypothesize that the responses for the Watch_UI_Act model has a higher (not significant) mean is that participants were interacting with their smart watch (which mostly just supports application notifications), and at that times, they were interruptible. This happened relatively infrequently as the watch was used relatively infrequently, thus not impacting the ESM score for Combo(x). Despite the lack of significance in these results, the results from the NASA-TLX survey clearly shows that users' workload perception was reduced by each of the different approaches in Attelia II. From these results we argue that, although users sometimes might not be aware of the direct value of Attelia II's breakpoint timing-based notification delivery, their overall workload perception was surely affected by our system.

Next, more detailed analysis on user's interruptibility with regard to changes in physical activity serves as an interesting research opportunity for us, given that smart phone platforms come with activity recognition APIs. Initially, we had a hypothesis that people would have breakpoints when changing from a "high energy" state to a "low energy" state, such as "stopping walking". However, our self-report data in Table 2 shows that there are breakpoints when going from low states to high states, such as "still" to "walking".

Classifying multiple levels of breakpoints is another opportunity to improve breakpoint detection capability of Attelia. Iqbal et al. found that there are at least three granularities of breakpoints, "coarse", "medium" and "fine", that users detect reliably [23]. Current Attelia implementation classifies a breakpoint in a binary manner, without considering granularity. Multi-level breakpoint detection, possibly using the information from the source detector, such as the concrete breakpoint type, points to an interesting future research opportunity.

Deployment of our system with "real" notifications from "real" Android applications is still an open challenge for us. Due to the current limitations with the Android platform, where the standard notification system can only be customized or replaced via OS-level changes, Attelia currently uses an artificial interruptive notification system we built for our evaluation. We are currently working on an "Attelia interruptibility API" on top of our middleware for other applications to use.

Further introduction and opportunistic combinations of other wearable devices, such as smart glasses [27] with gaze-tracking or blink recognition features, is yet another research opportunity for us. In users' multi-device situation, in which some of these devices have their own dedicated sensors and some do not, we need to investigate the value in having sub-sections of our system on the more impoverished devices.

Determining which device to send notifications is another research challenge. In our study, for simplicity, we configured the notification destination such that, if the user was actively manipulating the smart phone, then the notification was delivered on the phone. Otherwise, we defaulted to delivery on the watch. We may apply other display techniques developed by related research in the multi-display context [7, 10].

Currently, our "workload perception" measurement using nightly NASA-TLX survey has a limitation in terms of the temporal distances between the actual interruptive notification and the survey. Although we asked the participants to review their notification experience of the day in terms of the timings (not in terms of the number of notifications), their subjective evaluation at the end of the day can be influenced by several non-experiment-related aspects of their lives. Conducting such survey during the day itself can be another possible "workload" for the users. A lightweight but efficient survey methodology should be investigated as our future work.

CONCLUSION

In this paper, we proposed Attelia II, a novel middleware which detects opportune moments in which to interrupt and deliver notifications to users in their multi-device mobile and wearable environment. Based on UI-based and physical activity-based breakpoint detection techniques, Attelia II identifies such timings in real-time, without any external psycho-physiological sensors, and without modification to applications. Our evaluation through an extensive user study with smart watches and smart phones demonstrated the value of Attelia II. The addition of physical activity-based breakpoint detection improved Attelia's effectiveness over UI-based breakpoint detection. Attelia II also performed effectively on the smart watch. Our combinational breakpoint detection model that uses physical and UI-based breakpoint detections on both the watch and phone resulted in further significant reduction in users' workload perception.

ACKNOWLEDGEMENTS
We thank the anonymous reviewers and especially our shepherd for their help in improving the paper's presentation and technical content. The authors gratefully acknowledge the support of the students and staffs of Hide Tokuda Lab in Keio University and Ubicomp Lab in Carnegie Mellon University for continuous discussion, collaboration and advise as well as for helping to collect large amount of experimental data. This work was supported by Ministry of Education, Culture, Sports, Science and Technology (MEXT) Grant-in-Aid for the "Research and Development for Big Data Use and Application" and for the "Program for Leading Graduate Schools".

REFERENCES
1. Adamczyk, P. D., and Bailey, B. P. If not now, when?: the effects of interruption at different moments within task execution. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '04 (2004), 271–278.
2. Bailey, B. P., and Konstan, J. A. On the need for attention-aware systems: Measuring effects of interruption on task performance, error rate, and affective state. *Computers in Human Behavior 22*, 4 (2006), 685 – 708.
3. Begole, J. B., Matsakis, N. E., and Tang, J. C. Lilsys: Sensing unavailability. In *Proceedings of the 2004 ACM Conference on Computer Supported Cooperative Work*, CSCW '04 (2004), 511–514.
4. Böhmer, M., Lander, C., Gehring, S., Brumby, D. P., and Krüger, A. Interrupted by a phone call: Exploring designs for lowering the impact of call notifications for smartphone users. In *Proceedings of the 32nd Annual ACM Conference on Human Factors in Computing Systems*, CHI '14 (2014), 3045–3054.
5. Czerwinski, M., Cutrell, E., and Horvitz, E. Instant messaging: Effects of relevance and timing. In *People and computers XIV: Proceedings of HCI*, vol. 2, British Computer Society (2000), 71–76.
6. Czerwinski, M., Horvitz, E., and Wilhite, S. A diary study of task switching and interruptions. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '04 (2004), 175–182.
7. Dostal, J., Kristensson, P. O., and Quigley, A. Subtle gaze-dependent techniques for visualising display changes in multi-display environments. In *Proceedings of the 2013 International Conference on Intelligent User Interfaces*, IUI '13 (2013), 137–148.
8. Fischer, J. E., Greenhalgh, C., and Benford, S. Investigating episodes of mobile phone activity as indicators of opportune moments to deliver notifications. In *Proceedings of the 13th International Conference on Human Computer Interaction with Mobile Devices and Services*, MobileHCI '11 (2011), 181–190.
9. Garlan, D., Siewiorek, D., Smailagic, A., and Steenkiste, P. Project aura: toward distraction-free pervasive computing. *Pervasive Computing, IEEE 1*, 2 (april-june 2002), 22 –31.
10. Garrido, J. E., Penichet, V. M. R., Lozano, M. D., Quigley, A., and Kristensson, P. O. Awtoolkit: Attention-aware user interface widgets. In *Proceedings of the 2014 International Working Conference on Advanced Visual Interfaces*, AVI '14 (2014), 9–16.
11. Google Inc. Designing for accessibility - Android Developers. `https://developer.android.com/intl/ja/guide/topics/ui/accessibility/index.html`.
12. Google Inc. Making your app location-aware - Android Developers. `https://developer.android.com/intl/ja/training/location/index.html`.
13. Google Inc. The new multi-screen world — think with google. `http://www.google.com/think/research-studies/the-new-multi-screen-world-study.html`, Aug. 2012.
14. Gould, S., Brumby, D., Cox, A., González, V., Salvucci, D., and Taatgen, N. Multitasking and interruptions: a sig on bridging the gap between research on the micro and macro worlds. In *CHI'12 Extended Abstracts on Human Factors in Computing Systems* (2012), 1189–1192.
15. Haapalainen, E., Kim, S., Forlizzi, J. F., and Dey, A. K. Psycho-physiological measures for assessing cognitive load. In *Proceedings of the 12th ACM international conference on Ubiquitous computing*, Ubicomp '10 (2010), 301–310.
16. Hart, S. G., and Staveland, L. E. Development of NASA-TLX (task load index): Results of empirical and theoretical research. In *Human Mental Workload*, P. A. Hancock and N. Meshkati, Eds., vol. 52 of *Advances in Psychology*. North-Holland, 1988, 139 – 183.
17. Ho, J., and Intille, S. S. Using context-aware computing to reduce the perceived burden of interruptions from mobile devices. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '05 (2005), 909–918.
18. Horvitz, E., and Apacible, J. Learning and reasoning about interruption. In *Proceedings of the 5th International Conference on Multimodal Interfaces*, ICMI '03 (2003), 20–27.
19. Horvitz, E., Koch, P., and Apacible, J. Busybody: Creating and fielding personalized models of the cost of interruption. In *Proceedings of the 2004 ACM Conference on Computer Supported Cooperative Work*, CSCW '04 (2004), 507–510.
20. Hudson, S., Fogarty, J., Atkeson, C., Avrahami, D., Forlizzi, J., Kiesler, S., Lee, J., and Yang, J. Predicting human interruptibility with sensors: A wizard of oz feasibility study. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '03 (2003), 257–264.

21. Iqbal, S. T., and Bailey, B. P. Investigating the effectiveness of mental workload as a predictor of opportune moments for interruption. In *CHI '05 Extended Abstracts on Human Factors in Computing Systems*, CHI EA '05 (2005), 1489–1492.
22. Iqbal, S. T., and Bailey, B. P. Leveraging characteristics of task structure to predict the cost of interruption. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '06 (2006), 741–750.
23. Iqbal, S. T., and Bailey, B. P. Understanding and developing models for detecting and differentiating breakpoints during interactive tasks. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '07 (2007), 697–706.
24. Iqbal, S. T., and Bailey, B. P. Oasis: A framework for linking notification delivery to the perceptual structure of goal-directed tasks. *ACM Transactions on Computer-Human Interaction 17*, 4 (Dec. 2010), 15:1–15:28.
25. Iqbal, S. T., and Horvitz, E. Notifications and awareness: A field study of alert usage and preferences. In *Proceedings of the 2010 ACM Conference on Computer Supported Cooperative Work*, CSCW '10 (2010), 27–30.
26. Ishimaru, S., Kunze, K., Kise, K., Weppner, J., Dengel, A., Lukowicz, P., and Bulling, A. In the blink of an eye: Combining head motion and eye blink frequency for activity recognition with google glass. In *Proceedings of the 5th Augmented Human International Conference* (2014), 15:1–15:4.
27. JINS Co., ltd. JINS MEME. `https://www.jins-jp.com/jinsmeme/en/`, 2014.
28. Kahneman, D. *Attention and effort*. Prentice-Hall, Inc., 1973.
29. Kreifeldt, J. G., and McCarthy, M. E. Interruption as a test of the user-computer interface. In *JPL Proceeding of the 17 th Annual Conference on Manual Control* (1981), 655–667.
30. Machine Learning Group at the University of Waikato. Weka 3: Data mining software in java. `http://www.cs.waikato.ac.nz/ml/weka/`.
31. Müller, H., Pielot, M., and de Oliveira, R. Towards ambient notifications. *Peripheral Interaction: Embedding HCI in Everyday Life* (2013), 21.
32. Newtson, D., and Engquist, G. The perceptual organization of ongoing behavior. *Journal of Experimental Social Psychology 12*, 5 (1976), 436–450.
33. O'Conaill, B., and Frohlich, D. Timespace in the workplace: Dealing with interruptions. In *Conference Companion on Human Factors in Computing Systems*, CHI '95 (1995), 262–263.
34. Okoshi, T., Lu, Y., Vig, C., Lee, Y., Balan, R. K., and Misra, A. Queuevadis: Queuing analytics using smartphones. In *Proceedings of the 14th International Conference on Information Processing in Sensor Networks*, IPSN '15 (2015), 214–225.
35. Okoshi, T., Ramos, J., Nozaki, H., Nakazawa, J., Dey, A. K., and Tokuda, H. Attelia: Reducing user's cognitive load due to interruptive notifications on smart phones. In *Pervasive Computing and Communications (PerCom), 2015 IEEE International Conference on* (March 2015), 96–104.
36. Pejovic, V., and Musolesi, M. InterruptMe : Designing Intelligent Prompting Mechanisms for Pervasive Applications. In *Proceedings of the 2014 ACM International Joint Conference on Pervasive and Ubiquitous Computing*, UbiComp '14 (2014), 395–906.
37. salesforce.com. The 2014 mobile behavior report. `http://www.exacttarget.com/2014-mobile-behavior-report`.
38. Simon, H. A. Designing organizations for an information-rich world. *Computers, communication, and the public interest 37* (1971), 40–41.
39. Smith, J., and Dulay, N. Ringlearn: Long-term mitigation of disruptive smartphone interruptions. In *Pervasive Computing and Communications Workshops (PERCOM Workshops), 2014 IEEE International Conference on* (March 2014), 27–35.
40. Sony Mobile Communications Inc. SmartWatch 3 SWR50. `http://www.sonymobile.com/global-en/products/smartwear/smartwatch-3-swr50/`, 2014.
41. Speier, C., Valacich, J. S., and Vessey, I. The influence of task interruption on individual decision making: An information overload perspective. *Decision Sciences 30*, 2 (1999), 337–360.
42. Su, N. M., and Mark, G. Communication chains and multitasking. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '08 (2008), 83–92.
43. ter Hofte, G. H. H. Xensible interruptions from your mobile phone. In *Proceedings of the 9th International Conference on Human Computer Interaction with Mobile Devices and Services*, MobileHCI '07 (2007), 178–181.
44. Weiser, M., and Brown, J. S. The coming age of calm technology. In *Beyond calculation*. Springer, 1997, 75–85.
45. Zijlstra, F. R., Roe, R. A., Leonora, A. B., and Krediet, I. Temporal factors in mental work: Effects of interrupted activities. *Journal of Occupational and Organizational Psychology 72*, 2 (1999), 163–185.