

Institute for Visualization and Interactive Systems

University of Stuttgart
Universitätsstraße 38
D-70569 Stuttgart

Diploma Thesis Nr. 3683

Notifications in a Multi-Device Environment

Dominik Weber

Course of Study: Informatik

Examiner: Jun.-Prof. Dr. Niels Henze

Supervisor: Dr. Alireza Sahami

Commenced: July 16, 2014

Completed: January 15, 2015

CR-Classification: H.5.2

Abstract

Notifications are an integral part of how smartphones are used today. Apps can use notifications to inform the user about new text messages, upcoming events or available updates. By using visual cues, auditory signals and tactile output the attention of the user can be gained, enabling him or her to react right away. Notifications on desktop computers existed years before the introduction of smartphones and tablets are already widespread and share most of their characteristics. Other types of connected devices will soon join or already have joined smartphones in daily life. Recent development in the field of wearable devices indicates an upcoming widespread adoption of smartwatches and smartglasses. These devices can be used in situations where the use of a smartphone would be impractical or inappropriate. Previous work showed that notifications can distract the user, inducing stress and anxiety. Such effects could worsen with a growing number of notifying devices. Therefore, it is necessary to explore how notifications should behave in this “multi-device” environment. For that reason, we developed a concept for a framework that allows synchronizing notifications across multiple devices, including smartphones, tablets and desktop computers. Based on the approach of research in the large, we implemented the framework with support for a large number of devices as an update to an existing application and deployed it to a user base of several thousand users. After two months in the wild, the updated application was actively used by more than 33,000 users. On a per app basis the application allows disabling the content or sending notifications altogether. We analyzed the user settings for over 36,000 apps and found, for example, that apps from the “tools” category were excluded from sending notifications most frequently. On the other hand, the users disabled sending the text for apps related to communication most often. In a final large-scale study, the users rated the usefulness of synchronized notifications on different devices. Overall, synchronized notifications from messenger apps received the highest usefulness rating across all devices. The gained insights can support the development of ubiquitous notification mechanisms that keep users informed without overloading them.

Kurzfassung

Benachrichtigungen sind ein integraler Bestandteil bei der Nutzung von Smartphones. Apps können mithilfe von Benachrichtigungen den Benutzer über neue Nachrichten, anstehende Termine oder verfügbare Softwareaktualisierungen informieren. Die Aufmerksamkeit des Benutzers kann durch das Verwenden von visuellen Hinweisen, akustischen Signalen und taktilem Feedback gewonnen werden und ermöglicht dem Benutzer eine schnelle Reaktion auf die Benachrichtigung. Auf Desktop-Computern wurden Benachrichtigungen schon Jahre vor der Verbreitung von Smartphones eingesetzt. Auch Tablets sind bereits weit verbreitet und teilen sich viele Eigenschaften mit Smartphones. In absehbarer Zeit werden noch weitere Gerätetypen hinzukommen. Aktuelle Entwicklungen im Bereich des "Wearable Computing" deuten darauf hin, dass auch intelligente Uhren und Brillen bald verbreitet sein werden. Diese Geräte können in Situationen eingesetzt werden, in denen das Verwenden eines Smartphones unpraktisch oder unangemessen ist. Existierende Arbeiten zeigten, dass Benachrichtigungen den Benutzer von Aufgaben ablenken und Stress verursachen können. Mit einer wachsenden Anzahl von Geräten können diese negativen Effekte zunehmen. Daher sollte erforscht werden, wie sich Benachrichtigungen in einem Umfeld mit mehreren Geräten verhalten sollen. Aus diesem Grund wurde in dieser Arbeit ein Konzept für ein Framework entwickelt, das es ermöglicht, Benachrichtigungen über mehrere Geräte hinweg zu synchronisieren. Um das Framework im realen Umfeld mit einer großen Anzahl von Nutzern testen zu können, wurde eine bestehende Anwendung angepasst und die Änderungen wurden an eine bestehende Nutzerbasis mit mehreren tausend Benutzern verteilt. Nach zwei Monaten wurde die aktualisierte Anwendung von mehr als 33.000 Nutzern aktiv genutzt. Die Anwendung erlaubt es, für bestimmte Apps das Senden des Inhaltes oder die gesamte App auszunehmen. Die Analyse der Benutzereinstellungen für über 36.000 Apps ergab, dass Apps der Kategorie "Tools" am häufigsten vom Senden ausgenommen wurden. Andererseits wurde das Senden des Inhaltes meist für Apps deaktiviert, die im Zusammenhang mit Kommunikation stehen. In einer umfangreichen Studie haben die Benutzer die Nützlichkeit der synchronisierten Benachrichtigungen auf verschiedenen Geräten bewertet. Auf allen Geräten haben synchronisierte Benachrichtigungen von Messenger Apps die höchsten Bewertungen für die Nützlichkeit erhalten. Die gewonnenen Erkenntnisse können die Entwicklung eines ubiquitären Benachrichtigungssystems unterstützen, das den Benutzer informiert, ohne ihn oder sie zu überlasten.

List of Abbreviations

API Application Programming Interface

HTTP Hypertext Transfer Protocol

HTTPS HTTP Secure

ID Identifier

JSON JavaScript Object Notation

OS Operating System

PHP PHP: Hypertext Preprocessor

SQL Structured Query Language

URL Uniform Resource Locator

Contents

1. Introduction	13
2. Related Work and Background	15
2.1. Notifications and Interruptions	15
2.2. Mobile Notifications	16
2.3. Notification Modalities	18
2.4. Notifications in the Wild	19
2.4.1. Desktop Operating Systems	20
2.4.2. Mobile Operating Systems	21
2.4.3. Web and Web Browsers	23
2.5. Synchronizing Notifications across Devices	24
2.6. Recently Announced Devices	25
2.7. Summary	26
3. Concept of a Multi-Device Notification Framework	27
3.1. Starting Point	27
3.2. Architecture	28
3.3. Requirements	29
3.4. Summary	30
4. Device and Notification Studies	31
4.1. Device Distribution	31
4.1.1. Design	31
4.1.2. Results	32
4.2. Notifications on Multiple Devices	33
4.2.1. Design	34
4.2.2. Results	35
4.3. Discussion	37
5. Implementation of the Framework	39
5.1. Existing Implementation	39
5.2. Android App	40
5.2.1. Setup and Device Pairing	40
5.2.2. Receiving Notifications	43

5.2.3.	Dismissing Notifications	43
5.2.4.	App Settings	44
5.2.5.	Notification History	46
5.2.6.	Paired Devices and Settings	46
5.3.	Server Architecture	46
5.3.1.	Authentication	47
5.3.2.	Device Registration	48
5.3.3.	Event Broadcasting	48
5.4.	Google Chrome Extension	51
5.4.1.	Setup and Device Pairing	51
5.4.2.	Receiving Notifications	51
5.4.3.	Dismissing Notifications	52
5.5.	Extensibility	53
5.6.	Limitations	53
5.7.	Summary	54
6.	Deployment and Usage Studies	55
6.1.	Publishing	55
6.2.	Users and Devices	57
6.3.	Private and Disabled Apps	57
6.4.	Usefulness of Synchronized Notifications	59
6.4.1.	Design	59
6.4.2.	Results	60
6.5.	App List Update	62
6.6.	Summary	64
7.	Conclusions and Future Work	65
7.1.	Summary and Conclusions	65
7.2.	Future Work	66
A.	Appendix	69
A.1.	Package Names and Categories	69
A.2.	Survey about Notifications on Multiple Devices (Full Size)	71
A.3.	Most Frequently Disabled Apps	72
A.4.	Most Private Apps	73
A.5.	Device and Language Distribution	74
A.6.	Updated Most Frequently Disabled Apps (Smartphone)	78
A.7.	Updated Most Frequently Disabled Apps (Tablet)	79
A.8.	Updated Most Frequently Disabled Apps (Desktop)	80
	Bibliography	81

List of Figures

2.1. The Desktop Notifications app	17
2.2. Notifications in Windows 7	20
2.3. Notifications in Windows 8	20
2.4. Notifications in Android 5.0	22
2.5. Notifications in iOS 8	22
2.6. Notifications in Google Chrome	24
2.7. Notifications in Android Wear	25
3.1. The existing architecture	28
3.2. Concept of the new architecture	29
3.3. Exemplary data flow	30
4.1. Device ownership survey: Survey notification	32
4.2. Device ownership survey: Popup	33
4.3. Device ownership survey: Results	34
4.4. Survey about notifications on multiple devices: Survey	35
4.5. Survey about notifications on multiple devices: Results (combined)	36
4.6. Survey about notifications on multiple devices: Results (categorized)	37
5.1. Android app: Setup	41
5.2. Android app: Sign-in flow	42
5.3. Android app: Synchronized notifications	44
5.4. Android app: App settings and notification history	45
5.5. Android app: Connected devices and settings	47
5.6. Server: Event broadcasting	49
5.7. Chrome extension: Sign-in flow	52
6.1. Categories of disabled apps	58
6.2. Categories of apps in “Private Mode”	59
6.3. Usefulness of synchronized notifications: Android app	60
6.4. Usefulness of synchronized notifications: Google Chrome extension	61
6.5. Usefulness of synchronized notifications: Results	62
6.6. Updated app settings: User Interface	63
6.7. Updated app settings: Results	64

List of Listings

- 5.1. Exemplary JSON object for creating a notification. 50
- 5.2. Exemplary JSON object for removing a notification. 50

1. Introduction

Notifications are an integral part of how smartphones are used today. Apps can use notifications to inform the user about new text messages, upcoming events or available updates. By using visual cues, auditory signals and tactile output the attention of the user can be gained, enabling them to react right away. Smartphones are computing devices that are “always on”, “always connected” and “always with the user”, and therefore ideal devices for notifications. Notifications on desktop computers existed years before the introduction of smartphones and tablets are already widespread and share most of their characteristics. Other types of connected devices will soon join or already have joined smartphones in daily life. Recent development in the field of wearable devices indicates an upcoming widespread adoption of smartwatches and smartglasses. These devices can be used in situations where the use of a smartphone would be impractical or inappropriate. They are used either as standalone devices or as an extension of the smartphone and can also gain the user’s attention through different modalities. Depending on their situation at the time a user might prefer to be notified on a specific device or receive notifications of a particular kind on different kinds of devices.

In our prior work we conducted a large-scale assessment of mobile notifications [36]. To gather data in a realistic context we built a mobile application that shows notifications from smartphones on desktop computers. Based on previous approaches of research in the large [16, 17, 29] we decided to distribute the application in an app store. This enabled us to collect almost 200 million notifications from more than 40,000 unique users. By analyzing the collected data sets and conducting large-scale studies we were able to show which kinds of notifications were perceived as important by the users and how fast they acted on them.

Previous work showed that notifications can distract the user inducing stress and anxiety [6, 7, 26]. Such effects could worsen with a growing number of notifying devices. Therefore, it is necessary to explore how notifications should behave in this “multi-device” environment. The aim of this thesis is to gain a first insight into the user’s behavior and preferences with regard to notifications on multiple devices. Starting from the existing application, we develop a concept for a framework that allows synchronizing notifications across multiple devices. As continuation of our previous in-the-large approach we update our application to realize the conceptual framework for a number of devices. Furthermore, we distribute the update to the existing user base and analyze behavior and preferences of the users. On top of that we conduct multiple large-scale studies to consolidate our findings. The gained insights can support the development of ubiquitous notification mechanisms that keep users informed without overloading them.

Structure of this Thesis

This thesis is structured as follows: In chapter 2 we discuss the related work and background information regarding notifications. Based on this we present our concept for a multi-device notification framework in chapter 3. In chapter 4 we report the results of two studies. We conducted these studies to determine which kinds of devices are common and on which devices the participants would prefer to be notified on. Based on the results we describe the implementation of the multi-device notification framework in chapter 5. Afterwards, in chapter 6, we outline the process of distributing the implemented framework to the users. Furthermore we describe our method of data gathering and present the results of the data analysis. Finally, we draw conclusions from the results of this thesis and propose ideas for future work in chapter 7.

2. Related Work and Background

In this chapter we discuss the related work and background information on notifications. First we look at previous work about notifications and interruptions caused by notifications. We then discuss recent research regarding mobile notifications and different notification modalities. Afterwards we describe the implementations of notifications in current popular desktop and mobile operating systems. Finally we take a look at the current state of synchronizing notifications across devices and provide an overview of recently announced devices that are able to notify the user.

2.1. Notifications and Interruptions

Previous work on notifications and interruptions caused by notifications focused on the task performance in the work context. Czerwinski et al. conducted a diary study about task switching and interruptions [7]. The participants categorized their tasks by themselves and recorded interruptions. The results of the study showed that resuming an interrupted task is difficult and that longer tasks are interrupted more often. Czerwinski, Cutrell and Horvitz also explored the effects of notifications depending on the time and the type of task [6]. They could verify that notifications are generally harmful for the task performance. The results of their study show that notifications are more disruptive depending on the pace of the task. In a follow-up study the researchers found that users are more likely to forget the primary task goal if they get interrupted early during the task [5]. Adamczyk and Bailey also came to the conclusion that the negative effect notifications have on performance depends on the time of interruption [1]. Leiva et al. conducted a large-scale observational study [26]. The results show that interruptions caused by notifications can delay the completion of a task significantly.

Mark et al. investigated work without email in an empirical study [27]. According to the results of the study, not being available via email can be an advantage. Without interruptions the participants were able to focus on a task longer, which resulted in less multitasking and reduced stress. However the researchers questioned the long-term persistence of this effect. They also concluded that being able to contact people instantly is very useful in the work environment and giving this up should be carefully considered. Fischer et al. found that the content of a notification plays a large role in the perceived importance [11]. Iqbal and Horvitz conducted a field study on the use and perceived value of email notifications in the

workplace [24]. According to the researchers notifications provide passive awareness. In the study email notifications were turned off. For some users this increased the task performance, but other users interrupted their tasks in order to check for new emails. The researchers conclude that “users acknowledge notifications as disruptive, yet opt for them because of their perceived value in providing awareness” [24].

Karen Church and Rodrigo de Oliveira conducted an interview and a large-scale study to learn more about the popularity of the instant messaging app WhatsApp [4]. They compared WhatsApp to SMS and concluded that SMS is regarded as more formal, reliable and better for privacy. WhatsApp on the other hand is regarded as informal and is used in a more conversational style. In the interview some participants stated that they are concerned about the increasing amount of notifications and interruptions caused by them. Putting the phone into silent mode helps to cope with the message overload, however the researchers state that users that use WhatsApp and SMS for business-related communication do not silence their phones in order to not miss any important business communication.

2.2. Mobile Notifications

With the increasing adoption of smartphones the effect of mobile notifications became a compelling topic for research. Pielot, de Oliveira, Kwak and Oliver created a model to predict attentiveness to mobile instant messages [33]. They looked at the “last seen” status message that is shown in the instant messaging app WhatsApp and concluded that it’s not possible to derive the user’s attentiveness from the status. Thus they created a model with the goal of determining if an instant message will be seen within a few minutes of its receiving or not. The model categorizes the user’s attentiveness in “high” and “low” attention. The researchers conclude that it’s better to predict “low” attention but receive a fast response than the other way around. Also a compromise between accuracy of the classification and the accuracy of the “high” class has to be made. This model was found to be better at predicting the attentiveness of users than the “last seen” status message provided by WhatsApp. Pielot also created a model for call-availability prediction [31]. As part of a large-scale study, he published an Android app in the Google Play Store which collected data about call-availability to train the model.

Pielot, Church and de Oliveira conducted an in-situ study about mobile phone notifications [32]. The participants installed an app that logged notifications and user behavior for one week. The 15 participants received an average of 63.5 notifications per day. Most notifications were messenger and email notifications. Even if the phone was muted the users reacted within a few minutes to the notifications. According to the researchers the notifications create an increasing amount of negative emotions but the users feel connected. Most communication is asynchronous but social pressure forces the users to respond quickly. Receiving dozens or hundreds of notifications per day can be overwhelming because they are always with the user and affect private and work life.

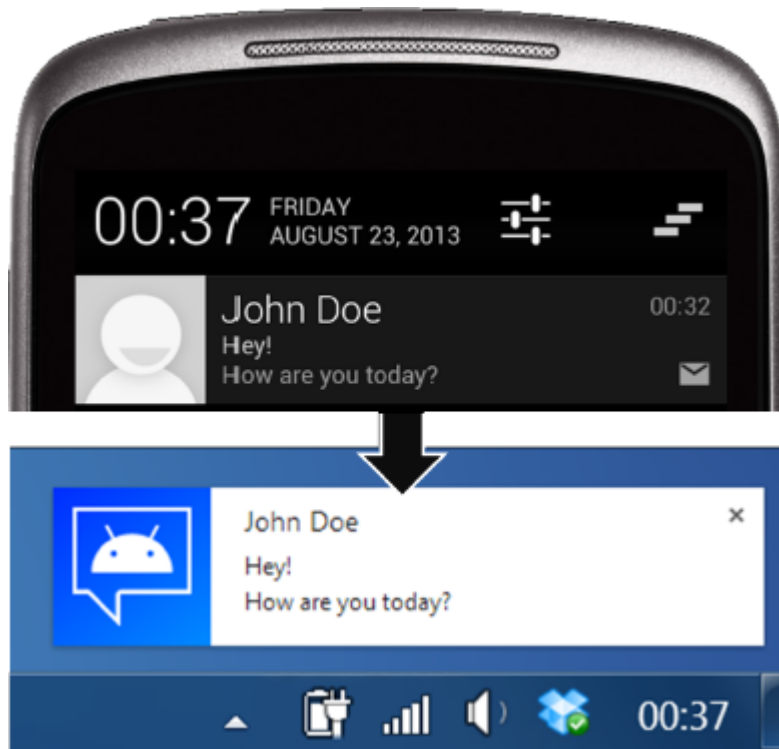


Figure 2.1.: The *Desktop Notifications* app forwards notifications from an Android device to the desktop [36].

In our own prior work we conducted the first large-scale analysis of mobile notifications [36]. We developed an Android app called *Desktop Notifications* which forwards notifications from an Android device to a desktop computer (see Figure 2.1). The notifications are routed through a server which enables us to record various data about notifications and user interaction. We used the app to collect data about the notifications but also subjective and qualitative feedback. We used this feedback to create a detailed analysis about which kind of notifications are liked and which are disliked. The app was published in the Google Play Store and can be downloaded for free. The users were informed in the store description that the app is part of a research project and that anonymous data would be collected. In addition to the Android app the user has to install a browser extension for Google Chrome or Mozilla Firefox. Because these browsers are available for all platforms, the app can be used regardless of the user's desktop operating system. The Android app detects new notifications on the device and sends the content and meta data of the notification to a server. The browser extension periodically polls the server to retrieve notifications. To pair devices the Android app generates a connection code which has to be entered into the browser extension. We collected almost 200 million notifications from more than 40,000 unique users. Additionally the time between the creation of a notification and the user clicking on the notification was recorded. The recorded click times showed that a notification is clicked on within the first 5 minutes with a probability of

83% and within the first 30 seconds with a probability of 50%. We concluded the analysis with guidelines for designers to make effective use of notifications in mobile apps. Notifications related to messaging, people and events are perceived as important by the users, while system notifications are perceived as less important.

In a study conducted by Mashhadi et al. the notifications of 10 participants were logged over the course of 15 days [28]. In interviews after the study the participants told the researchers that a fine-grained notification management is important.

The website *Wired* published an article with the title “Why Notifications Are About to Rule the Smartphone Interface” [20]. According to the article the notification screen is the most important screen on a smartphone, because it doesn’t only notify, but it also is interactive. When receiving a new instant message the user can tap the notification to open the app or even reply directly from the notification screen, which is an important aspect of asynchronous communication. The article further states that interactive notifications are more useful than static notifications because the user doesn’t even have to open the app.

2.3. Notification Modalities

To gain the user’s attention typically visual cues, auditory signals and tactile output are used. Visual cues include the use of light-emitting diodes (LEDs), pop-ups, movement, blinking, icons and badges. Sound notifications use different sounds and melodies. Tactile output makes use of vibration motors primarily found in mobile devices and can vibrate in different patterns and multiple levels of intensity. These notification channels can be used together or separately from each other. For example an app on a mobile phone can inform the user about a new message by playing a sound, vibrating and showing parts of the message at the top of the screen. However the user might also decide to silence the phone, so only the tactile output and visual cues are used. Hansson, Ljungstrand and Redström studied these notifications channels and built a model for subtle and public notifications for mobile devices [14]. The model classifies various notification cues according to their subtlety and publicity. The researchers state that auditory notifications are intrusive while tactile cues are private and subtle. Furthermore they explain that in some situations the lack of public notification cues might create awkward situations. They give an example of a conversation between two persons where one person receives a notification through a subtle notification cue and is suddenly distracted from the conversation without the other person noticing. The Reminder Bracelet by Hansson and Ljungstrand [13] introduces the idea of a wearable device that is worn on the wrist. The device is connected to a personal digital assistant (PDA) and can inform the user about events through visual cues. The evaluations of studies conducted by the researchers revealed that the participants preferred these subtle cues to the public auditory cues of the PDA.

There has been extensive research regarding the expansion of the common notification modalities. For example Vibkinesis, a smartphone proposed by Yamanaka and Miyashita [41], can

inform the user about new messages by changing its rotation. By using several vibration motors it is possible to rotate and move the device. An example use case given by the researchers is changing the orientation of the device to inform the user about a new message. Because the orientation stays the same if the battery of the device dies the notification persists when other notification modalities would cease to function. The researchers also added an omni-directional lens to the device, which makes it possible to sense the surrounding area to detect and move towards the user. Sahami et al. proposed using multiple vibration actuators in a mobile phone to enhance the tactile feedback [35]. In experiments they added up to six vibration motors to phone prototypes, and thus enhancing the typical vibration pattern and variation of the vibration intensity by the location of the active actuators. Based on their findings they concluded that the actuators should be placed in the corners of the device and that the position of the actuators can be used to encode information.

How much information can be encoded in a three light visual display and how fast the users are able to learn it was studied by Campbell and Tarasewich [3]. In a follow-up study Tarasewich et al. tested notification cues consisting of three multicolored lights preceded by vibration on a handheld device [37]. The results show that customizing notification cues enhances learning and usefulness of notifications. Furthermore the tactile feedback helped to increase awareness for new notifications.

Gomes et al. created the MorePhone [12]. The MorePhone is a flexible bendable smartphone. The researchers evaluated the usefulness of the actuation. The results show that for important notifications, like alarms and voice calls, the whole display should be actuated. For less important notifications only the corners should be actuated – preferably a certain corner for each type of notification. Another actuated system was created by Hemmert et al. [15]. They created a prototype of a mobile phone with an actuated back plate which allows two-dimensional tapering.

Warnock et al. conducted a study about notification modalities with older users. The researchers concluded that older users were affected by the same modalities as younger users [39].

2.4. Notifications in the Wild

In the following sections we describe the current implementation of notifications in the most popular operating systems. We first look at the implementation of notifications in the desktop operating systems Windows and OS X. Afterwards we describe notifications on the mobile operating systems Android and iOS. Finally we discuss the current state of notifications in the web and in web browsers.

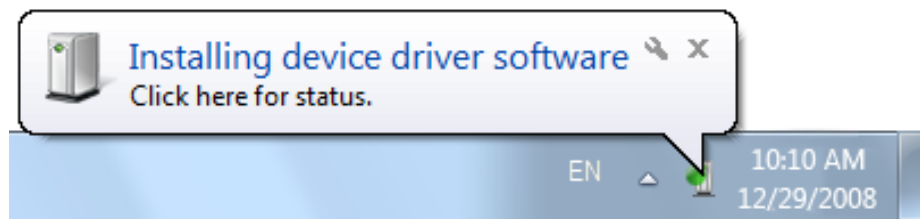


Figure 2.2.: A balloon-style notification in Windows 7^a.

^aImage from the Microsoft Developer Network <http://msdn.microsoft.com/en-us/library/dn742495.aspx>

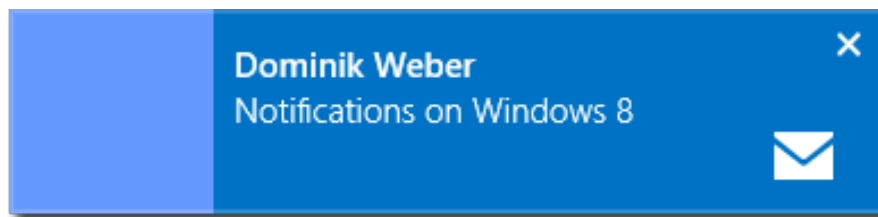


Figure 2.3.: A Windows 8 email notification.

2.4.1. Desktop Operating Systems

In desktop operating systems notifications are typically shown near the system tray area. In Microsoft Windows this is the bottom right corner of the screen by default. Figure 2.2 shows a balloon-style notification in Windows 7. The notification consists of an icon, a title and an additional line of text. The notification is actionable by clicking on it. There is also an “x” icon to close the notification and a wrench icon to disable notifications for the program in question. In this example the notification originates from the icon in the system tray. As result of many applications implementing the notifications themselves, various notification styles exist. With Windows 8 Microsoft tried to create an unified look and feel across the system, including notifications. In Figure 2.3 a notification from the default email app is shown. The notification uses the prominent app color (in this example blue) as background and allows the display of an image, a title and an additional line of text. The title is used to display the sender name and the additional line for the subject of the mail. The image is used to display a contact image of the sender – in this example a blue square. There is an additional icon to indicate which app created the notification. The notification is actionable, clicking or tapping on it will open the newly received email. Depending on the input mechanism the notification can be closed by clicking on the “x” or swiping it away. On all Windows versions the notifications fade out after a couple of seconds and there is no way to get them back. Windows 10, which at the time of writing is available as a preview version, will include a notification center that contains all active notifications until dismissed.

In Apple's desktop operating system OS X notifications are shown in the top right. The notifications can have an icon, title and content and are actionable. Optionally button and text input fields can be attached to them. They, too, fade out after a couple of seconds. In OS X 10.8 (Mountain Lion) Apple introduced a notification center which displays active notifications in a unified manner. The notification center can also hold widgets like weather and stock information.

2.4.2. Mobile Operating Systems

In the Android operating system notifications are a core feature and were available since the first public version. Every Android app can use the notification API to alert the user. Figure 2.4 shows how notifications look in Android 5.0. A notification can play a notification sound, let the device vibrate in a specific pattern and display an icon in the top left of the screen. In previous versions of Android a so called ticker text could temporarily replace the status area at the top of the screen with the text of the notification, which allowed the user to glance at the content. In Android 5.0 this ticker was replaced with *heads up* notifications that temporarily cover the top part of the screen, similar to notifications on desktop operating systems. Also present as a feature in the first version of Android was the notification center. It is always accessible by swiping down from the top of the screen and reveals a list of active notifications. A basic Android notification consists of a small icon, a title and a content text. These basic notifications can be enhanced in multiple ways. In Android 4.1 rich notification layouts were introduced which allow replacing the single line of text with multiple lines of text, a progress bar, a full size image or list items. Also the small icon can be replaced by a larger image which is typically used to display contact images next to messages. Developers can set a specific action for when the user taps on the notifications. If more options are needed it is also possible to add action buttons below the notification. For even more options it is possible to implement a custom view, which for example can be used to display media control elements. Notifications on Android can be dismissed by swiping them to the left or right. There is also a button which clears all notifications at once. With Android 5.0 active notifications are also displayed on the lock screen of the device, making them ubiquitous and easily accessible from all screens.

Notifications in the iOS operating system started as badges on app icons. As seen in the bottom right of Figure 2.5 a badge is a red circle with a number, indicating the number of new elements. Later alert notifications were added, which are modal dialogs that pop up in the center of the screen and require the user to either take action or dismiss them. The less intrusive banner notifications were added afterwards. Banner notifications appear at the top of the screen and can be dismissed by swiping them away. Both alert and banner style notifications can display two rows of text and action buttons. Banner notifications can additionally display an icon next to the text. The user can decide for every app which style should be used, or disable notifications altogether. On iOS notifications are shown on the lock screen. With iOS 7 a notification center, like the one found on Android, was added to the operating system. The

2. Related Work and Background

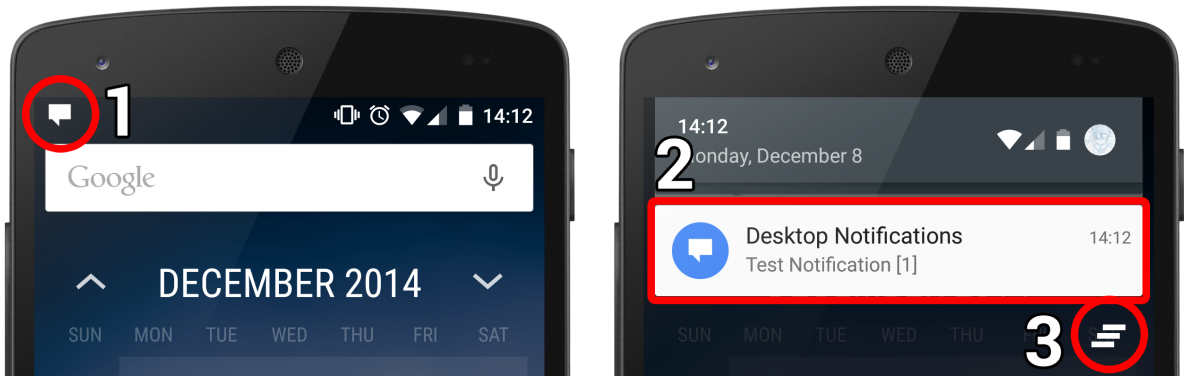


Figure 2.4.: Notifications on Android 5.0. — (1) Notification icon. (2) The notification. (3) “Clear all” notifications button.

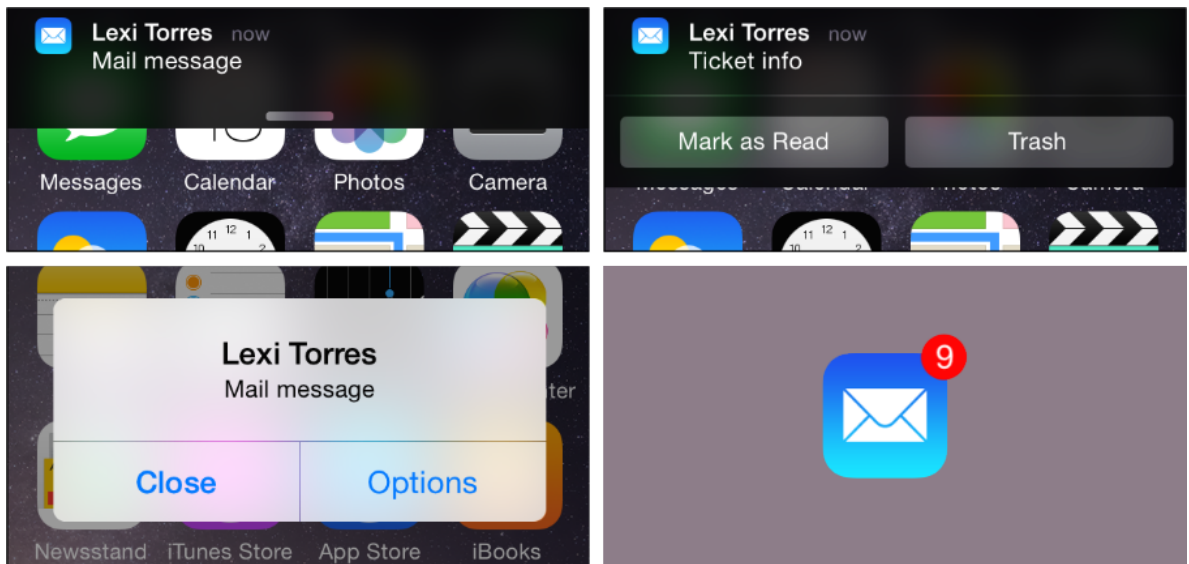


Figure 2.5.: Notifications on iOS 8^a. — *Top left:* Banner notification. *Top right:* Banner notifications with two action buttons. *Bottom left:* Alert notification. *Bottom right:* Notification badge.

^aImages from the iOS Developer Documentation <https://developer.apple.com/library/ios/documentation/UserExperience/Conceptual/MobileHIG/NotificationCenter.html>

notifications on iOS evolved from simple badges to rich notifications with multiple actions that can be accessed from any screen by swiping down from the top of the screen.

2.4.3. Web and Web Browsers

To enable a platform-independent approach of notifications the Web Notification [38] specification was added to the HTML5 [2] standard. The goal of Web Notifications is to allow websites to notify the user. The specification is still a working draft at the time of writing, however the API is already partially supported by desktop browsers. All major browsers except the Microsoft Internet Explorer support the API without any vendor prefixes. To prevent abuse a website first has to request permission to show notifications. If the permission is granted a simple method call is enough to display notifications. The specification requires an icon, title and content row. Optionally, the notifications can be made actionable. However it is not part of the specification how the notifications should actually be displayed. The Safari and Opera browsers use the underlying functionality of the host operating system to display the notifications. In Firefox and Chrome the notifications are implemented platform-independent. Unlike the other implementations Chrome has a notification center which integrates in the form of a bell icon in the system's tray area and contains all active notifications. All browsers respect the default position of notifications in their host operating system. On Windows browser notifications are shown in the bottom right, on OS X in the top right. The support for web notifications by mobile browsers is still limited¹.

One limitation of Web Notifications is that the website has to be opened in the browser in order to notify the user. A solution to this challenge are Service Workers [34]. This specification, which currently is also a working draft, aims to define persistent background scripts that run even when the corresponding website is not open. The background scripts can receive data from a server and then use Web Notifications to notify the user.

Most web browsers can be extended with browser extensions. Several browser vendors added a notification API that can be used by extensions. These APIs are different from the Web Notification API because they don't require permission to display notifications. In Firefox the notifications created by add-ons look and feel the same as Web Notifications. In Chrome basic notifications created by the extension API also look the same, however they can be extended in various ways. Figure 2.6 shows an active notification in Chrome's notification center. Apart from the usual icon, title and content there is also a third line of text and a button. It's possible to add up to two buttons to every notification. In addition to this basic style it's possible to include an image that spans the whole width of the notification, a notification containing a text list, and a progress bar. The notification center itself integrates into the host operating system with a bell icon in the system's tray area. Similar to Android all active notifications can be dismissed at once by clicking on the "clear all" button.

¹Browser support for Web Notifications <http://caniuse.com/#feat=notifications>

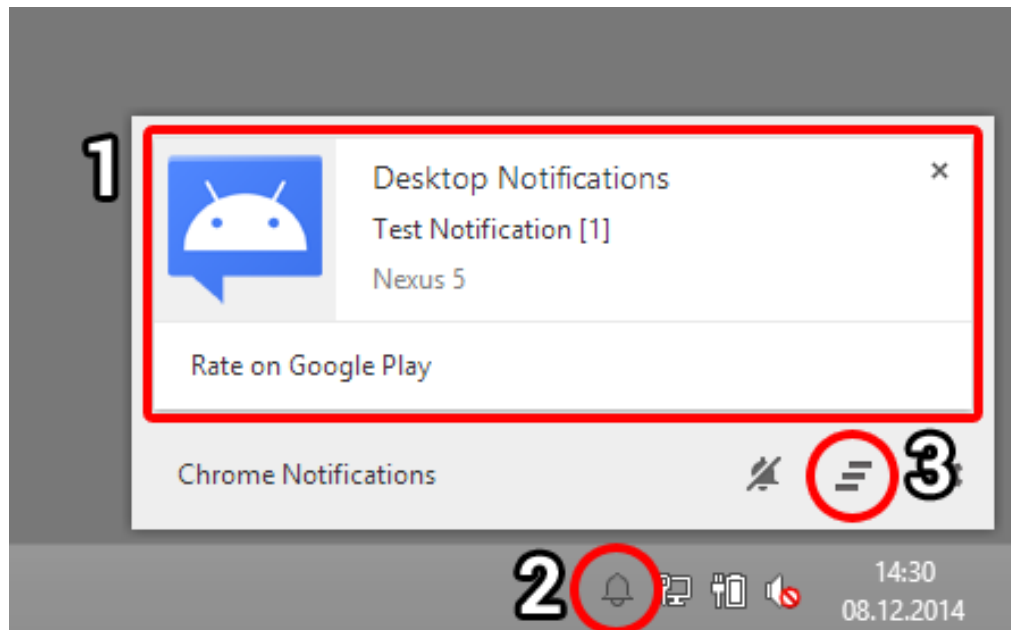


Figure 2.6.: The notification center of Google Chrome with one active notification. — (1) The notification. (2) Notification indicator. (3) Icon to dismiss all notifications.

2.5. Synchronizing Notifications across Devices

Typically notifications are only shown on the device on which the app or program that created the notifications is installed. If the user wants to receive notifications on all devices, for example about new emails, he has to set up his account on every device. In some cases this isn't possible. For example the popular messenger WhatsApp can only be installed on one device at a time, as it is bound to the phone number. On the other hand email clients are available for all platforms. However those are not always smart about notifying the user. A notification for a newly received email might be shown on all devices with the client, but not all clients remove the notification once the user read the email. With the increasing amount of connected devices, dismissing the same notification on multiple devices can become an annoyance. Because of this some social networking services and messengers started to synchronize the state of notifications across multiple devices. In 2013 Apple introduced a feature which allows some notifications from the iPhone to be shown on Macintosh computers. The notifications are sent to the computer via Bluetooth. Similarly some wearable devices like smartwatches and advanced fitness trackers also connect to the user's smartphone and are able to notify the user about new events. In December 2012 we created the *Desktop Notifications* Android app [36] which allows sending notifications from an Android smartphone to the user's desktop computer.

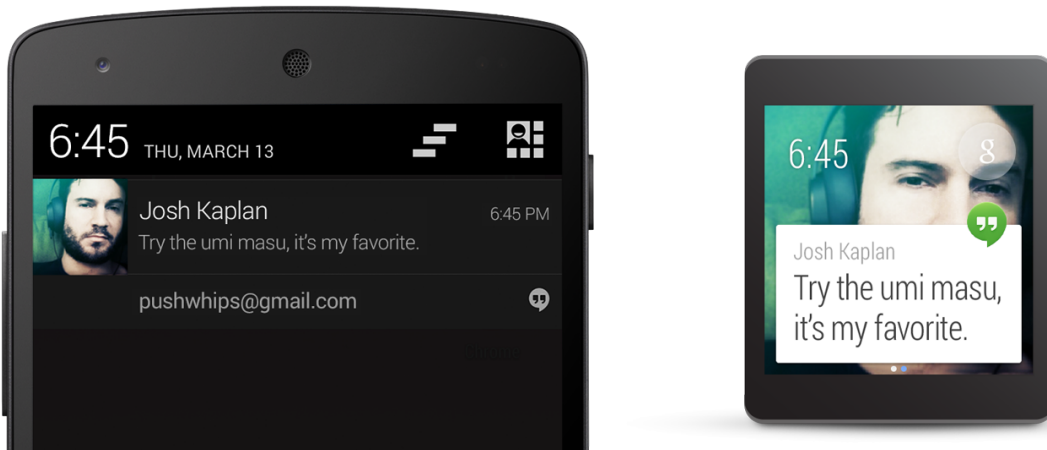


Figure 2.7.: A notification displayed on both an Android-based smartphone and an Android-Wear-based smartwatch^a.

^aImage from the Android Wear Developer Documentation <https://developer.android.com/training/wearables/notifications/index.html>

2.6. Recently Announced Devices

As this thesis was written, several new devices with the capability to notify the user have been announced or released. Apple unveiled the Apple Watch², a smartwatch that connects with iPhones. One focus of the presentation was the ability to show the phone's notifications on the watch, including messages, calls and notifications from third-party apps. The notifications on the watch are interactive, so it's possible to accept calls and respond to text messages. While the Apple Watch only has been announced, several other manufacturers have already started to ship smartwatches based on the Android Wear³ operating system. Android Wear connects with Android-based smartphones and automatically shows all notifications from the smartphone on the smartwatch as well (see Figure 2.7). Dismissing a notification on either device also removes it from the other device. This functionality works with existing apps, however the notifications on the smartwatch can be enhanced by adding special actions or graphics. The smartglasses Google Glass use a modified version of Android rather than Android Wear, and the ability to synchronize notifications with a smartphone is a feature that was previously absent. However Google announced⁴ that Glass will soon be able to display all notifications from the connected smartphone on Glass. Pebble, one of the first smartwatches, received a firmware update⁵

²Apple Watch <https://www.apple.com/watch/>

³Android Wear <http://www.android.com/wear/>

⁴Notifications on Google Glass (Announcement) <https://plus.google.com/+GoogleGlass/posts/FSfaTv9yfb>

⁵Notifications on the Pebble Smartwatch (Announcement) <https://blog.getpebble.com/2014/11/20/fw28-adr21-ios252/>

which also enabled showing notifications from any app on the watch. Microsoft released the fitness tracker Microsoft Band⁶ that can also be paired with a smartphone. Unlike the Apple Watch or Android-Wear-based smartwatches it supports all major mobile operating systems (iOS, Android and Windows Phone). The device was also advertised to be able to synchronize notifications with the phone. Furthermore Microsoft also announced a smart wireless charging pad and a smart lamp that can notify the user via ambient light⁷.

2.7. Summary

In this chapter we looked at related work regarding notifications and interruptions caused by notifications. Studies have shown that notifications are disruptive and lower task performance [6, 7, 26] but are still valued by the users [24]. We then looked at recent research regarding mobile notifications. The *Desktop Notifications* [36] app allows sending notifications from Android-based devices to the desktop. Afterwards we described different notification modalities and related research. Furthermore we discussed the current implementation of notifications in popular desktop and mobile operating systems, notifications in the web and the state of synchronizing notifications across devices. Finally we concluded the chapter with a selection of (mostly wearable) devices that are able to notify the user and were announced or released during the work on this thesis.

⁶Microsoft Band <http://www.microsoft.com/microsoft-band/en-us>

⁷Microsoft Smart Charging Pad and Smart Lamp <http://lumiaconversations.microsoft.com/2014/11/12/smarter-charging-notifications-new-light/>

3. Concept of a Multi-Device Notification Framework

In this chapter we introduce the concept of a multi-device notification framework. The purpose of the framework is to synchronize notifications across all kinds of devices by expanding an existing application. First we explain the architecture of the existing system. We then outline the requirements of the framework and the new architecture.

3.1. Starting Point

We use the *Desktop Notifications* [36] app as starting point for our concept. The app allows the sending of notifications from an Android-based device to the desktop. Figure 3.1 shows the architecture in a simplified manner. The sender device shown in the figure is an Android smartphone or tablet with the *Desktop Notifications* app installed. The app listens for new notifications on the device and sends them to a central server. The server temporarily stores these notifications in a database. On the receiving device, a desktop computer, the user has to install an extension for either the Google Chrome or the Mozilla Firefox browser. The browser extension periodically queries the server for new notifications sent by the Android app. Multiple receiving devices can query the server at the same time, thus creating a one-to-many relationship between the sender and the receiving devices. If the server returns a notification, the browser extensions displays it on the desktop.

This architecture has several limitations. First, there is a distinction between the sending and receiving devices and the data only flows in one direction. The architecture does not allow sending notifications from one sender device to another. Furthermore, notifications have to be dismissed on each device individually. Also the receiving devices have to know the sender device in order to query new notifications from the server. For example, if a user owns a smartphone and tablet, the receiving devices have to be paired explicitly with each sender device. It would be advantageous to be able to automatically pair devices with the knowledge that both devices belong to the same user. From a data flow perspective, periodically querying the server can create a lot of unnecessary requests. Furthermore, the delivery of notifications to the receiving devices may be delayed, as the length of time between two requests is a balancing act between minimizing that delay by querying for new notifications frequently and minimizing the workload of the server.

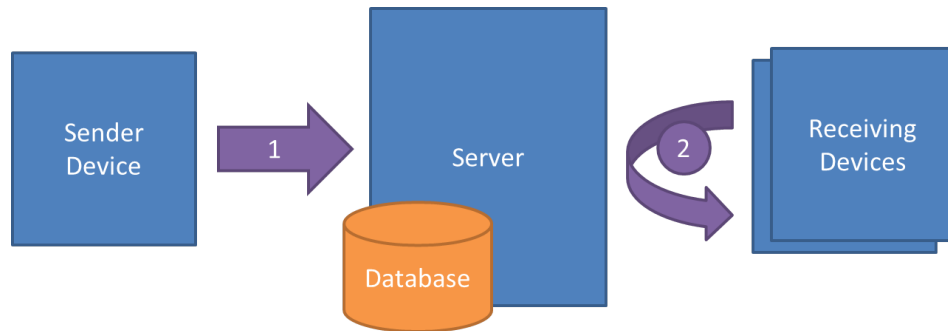


Figure 3.1.: The architecture of the *Desktop Notifications* [36] app that we used as a starting point. — (1) The sender device sends information about new notifications to the server. (2) The receiving devices periodically query the server for new notifications.

3.2. Architecture

Based on the existing architecture we propose a notification framework for a multi-device environment. As shown in Figure 3.2, the framework still uses a central server as its core element. In a study regarding the risk associated with permissions on mobile phones it was shown that sharing information with a server was regarded as not critical [9]. Instead of distinguishing between sender and receiver devices, every device now takes on both roles. Furthermore every device is associated with a certain user. The devices itself no longer need to know about each other; they now only need to know which user they belong to. The central server is used to keep track of all user-device-relations. This way the server can retrieve a list of all devices for a given user. The devices communicate with the server via a bi-directional channel. This communication channel does not necessarily have to be persistent, as long as the devices can send data to the server at any time and vice versa. With the server able to push messages to the devices, it is possible to broadcast notifications with very little delay to all connected devices. In addition to sending notifications to all devices, the dismissal of notifications is now also broadcasted. Figure 3.3 shows an exemplary data flow between three devices owned by the same user and the server. In step (1) a notification is shown on the first device and information about the notification is sent to the central server. The server knows which other devices belong to the user and forwards the notification in step (2) to the second and third device. Both devices then notify the user based on the received information. At this point the notification is shown on all devices. In step (3) the user dismisses the notification from the second device. The device informs the server about the dismissal, which in return broadcasts a remove event to the first and second device. Both devices remove the notification and thus the notification is cleared on all devices at once with minimal delay. In the following chapters we will refer to the notification of step (1) as the notification from the device of origin and the notifications created on the other devices as synchronized notifications.

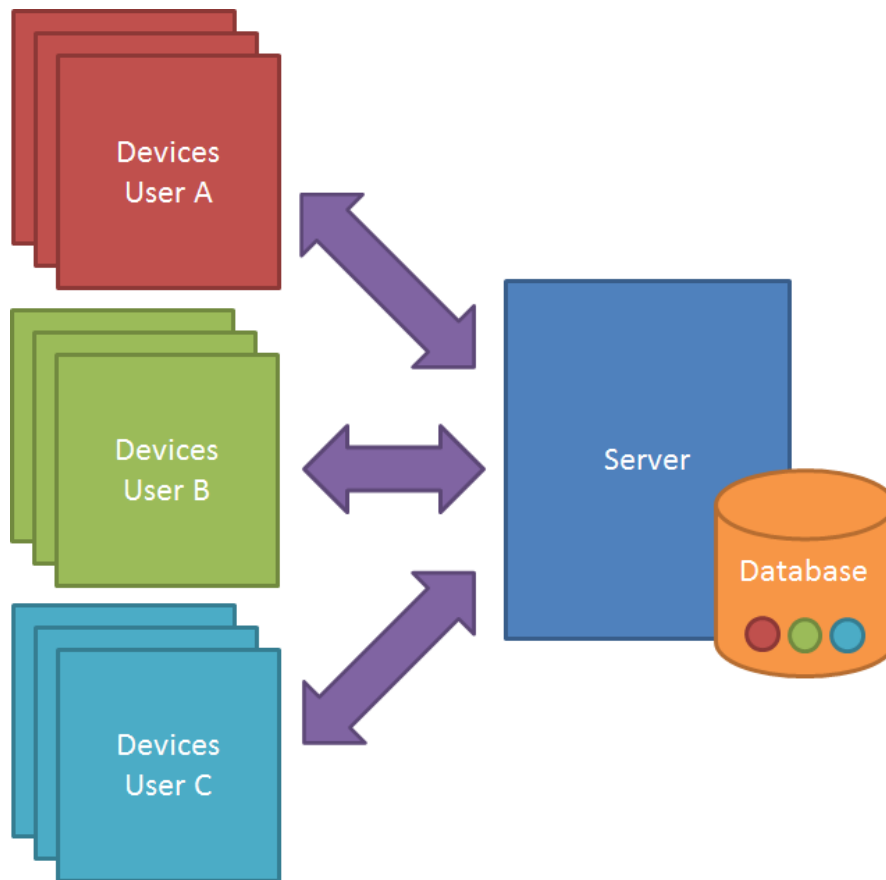


Figure 3.2.: Concept of the multi-device notification framework. The user’s devices are connected to a central server via a bi-directional, but not necessarily persistent, communication channel. The server keeps track of the devices by storing the user-device-relations in a database.

3.3. Requirements

The new architecture imposes several requirements. Firstly, to include a device within the framework it must be able to communicate with the server in some way. For mobile devices this communication channel should not impact the battery. Additionally, the server should be able to push events to the devices. Furthermore a device has to be able to display notifications. As we have shown in section 2.4 most notifications nowadays consist of an icon and two lines of text. That means that in order to synchronize notifications across devices at least the text has to be transmitted. On top of that there should be an easy way to assign devices to a certain user and store this relation on the server.

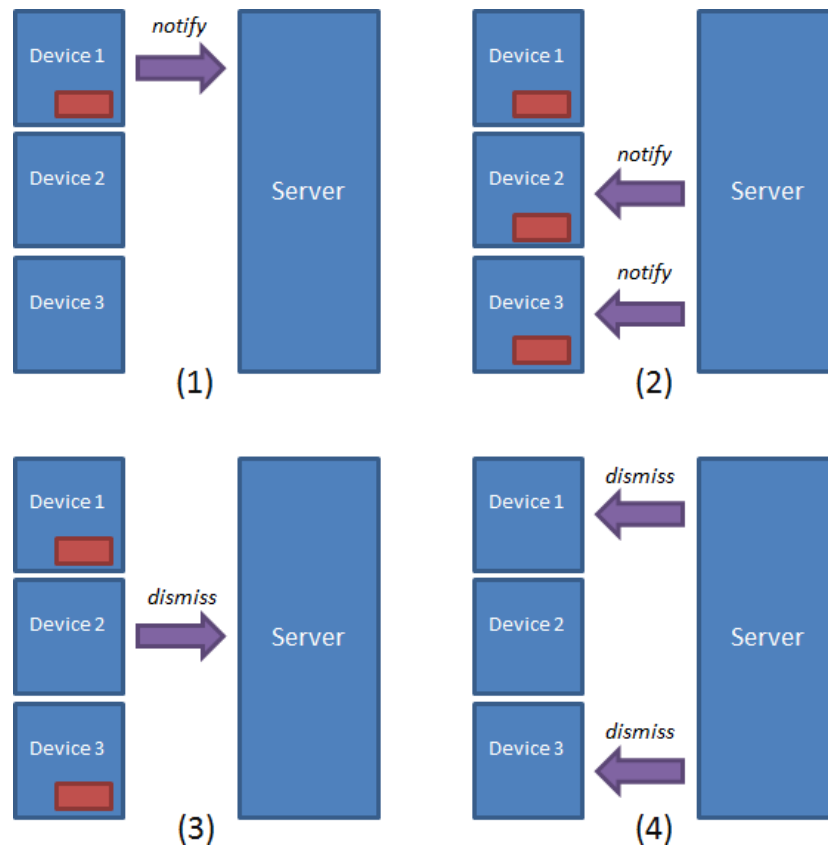


Figure 3.3.: Exemplary data flow between three devices owned by the same user and the server. A notification is sent from the first device to the two others via the server. The user then dismisses the notification on the second device, which causes the notifications to disappear from all devices in short order.

3.4. Summary

In this chapter we proposed a framework for notifications in a multi-device environment. A central server keeps track of the user's devices and is able to communicate with them over a bi-directional communication channel. An application is installed on each of the devices and listens for notifications. If a new notification is registered, the application forwards the notification to the server, which in return broadcasts it to the user's other devices. If the user dismisses a notification on one device, the dismissal is also broadcasted to all devices, thus removing the notification on all other devices in short order.

4. Device and Notification Studies

To learn more about notifications on multiple devices we conducted two studies. For both studies we pushed surveys to the users of the *Desktop Notifications* [36] app. We first asked the users what kind of devices they own. In the second survey we asked them on which devices they prefer to be notified. In this chapter we explain the design and results of the surveys, and discuss the findings.

4.1. Device Distribution

The goal of the first survey was to determine the device distribution among the users of the *Desktop Notifications* app in order to get a better understanding of the user base. We asked the users which kinds of devices they own.

4.1.1. Design

Similar to the large-scale study conducted in [36], our approach was to show this survey to the users of the *Desktop Notifications* extension for Google Chrome. We modified the Chrome extension to display a survey notification to every user. As shown in Figure 4.1, we asked the user “What kind of devices do you own?”. Below the notification we added two buttons. The first button (*Answer*) opened the survey in a pop-up window and the second button (*I don’t want to answer*) hid the notification and set a flag so that the user was not asked again. The survey notification was shown once every day until the user either agreed to participate or declined.

Figure 4.2 shows the pop-up window which was displayed to the user if he decided to participate in the survey. In the pop-up window the question was slightly modified to “Which of the following devices do you own?”, followed by a list of seven device categories: *Smartphone*, *Tablet*, *Smartwatch*, *Smartglasses*, *Desktop PC*, *Laptop* and *TV*. We chose these device categories by looking at currently available devices and categorizing them. The individual categories could be selected by marking checkboxes. For the categories *Smartwatch* and *Smartglasses* we added exemplary devices. The categories were listed in a random order. Below the list the participants could optionally add a text comment. We translated the survey into English and German and detected the set language of the Chrome installation to determine which should

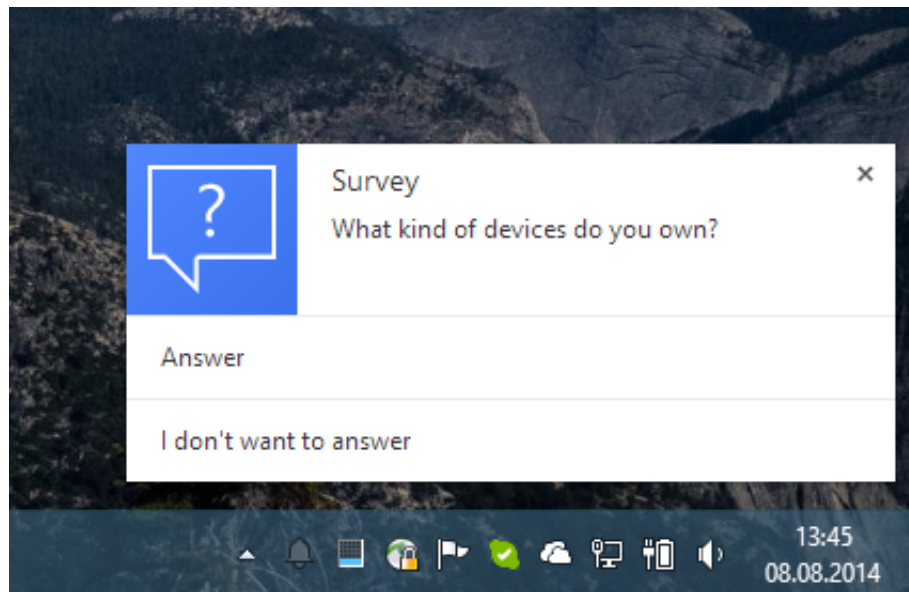
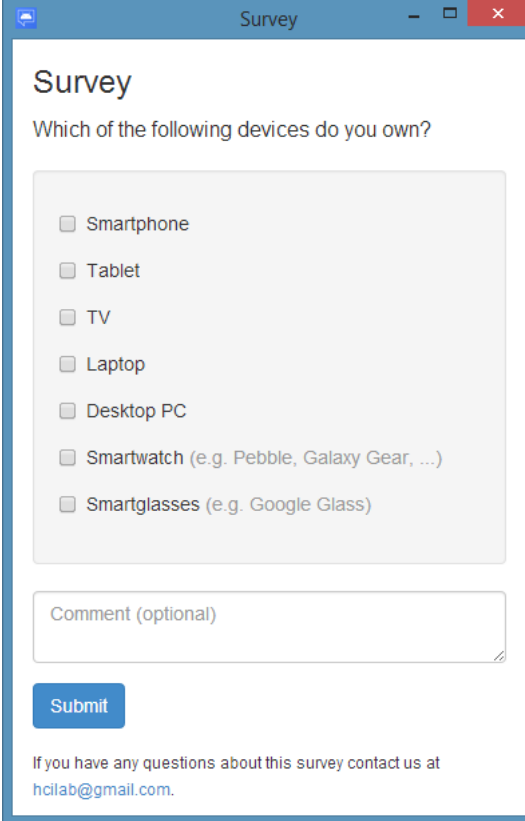


Figure 4.1.: We asked the users to participate in the survey by displaying a notification on the user's desktop.

be displayed. For German users we displayed the German version and the English one in all other cases.

4.1.2. Results

The survey was run over the course of four weeks. In total 6,779 users participated. Even though the survey was only shown in English and German, we used the data from all participants, regardless of their set language. The most popular device languages were English (45.34%), German (11.95%), Portuguese (9.97%), Spanish (6.78%) and Japanese (4.22%). The complete language distribution is available in appendix A.5. Figure 4.3 shows the results of the survey. 92% of the participants own a smartphone. Laptops and PCs are owned by 61% and 53%, respectively. These high numbers are not surprising because the participants are users of the *Desktop Notifications* app, which requires an Android smartphone or tablet and a desktop PC or laptop to work. Next are TV (45%) and tablet (44%). An interesting thing to look at is the smartphone-tablet-distribution. Only 2% of the users own a tablet, but no smartphone, and 42% own both. When we conducted the survey, there was a limited number of smartwatches available for purchase. 8% of the users owned a smartwatch. Even less popular are smartglasses, the select of which is pretty much limited to Google Glass. We expected many users to not even know about them, so the result of 2% for the smartglasses ownership is not at all surprising. In the comments some users told us that they plan to buy a smartwatch, but are waiting for the technology to mature. It's important to note that we did not ask if the TV in



The image shows a browser window titled "Survey" with a blue header bar. The main content area has the title "Survey" and the question "Which of the following devices do you own?". Below the question is a list of seven device categories, each with an unchecked checkbox: Smartphone, Tablet, TV, Laptop, Desktop PC, Smartwatch (e.g. Pebble, Galaxy Gear, ...), and Smartglasses (e.g. Google Glass). Underneath the list is a text input field labeled "Comment (optional)". At the bottom left of the form is a blue "Submit" button. At the bottom of the window, there is a small text block: "If you have any questions about this survey contact us at hcilab@gmail.com."

Figure 4.2.: The survey shown to the users of the *Desktop Notifications* extension for Google Chrome. We asked the participants what kind of devices they own, with a list of predefined categories and an optional comment field.

question was capable of connecting to the Internet. Some users stated in the comments that their TV is indeed not a “smart” TV. Others stated that they use gaming consoles and devices like the Google Chromecast to access online content on their TVs.

4.2. Notifications on Multiple Devices

After determining the device distribution among the *Desktop Notifications* users we wanted to learn more about which devices the users preferred to be notified on. Thus we conducted a second survey asking the users on which devices they would like to see notifications from certain apps.

4. Device and Notification Studies

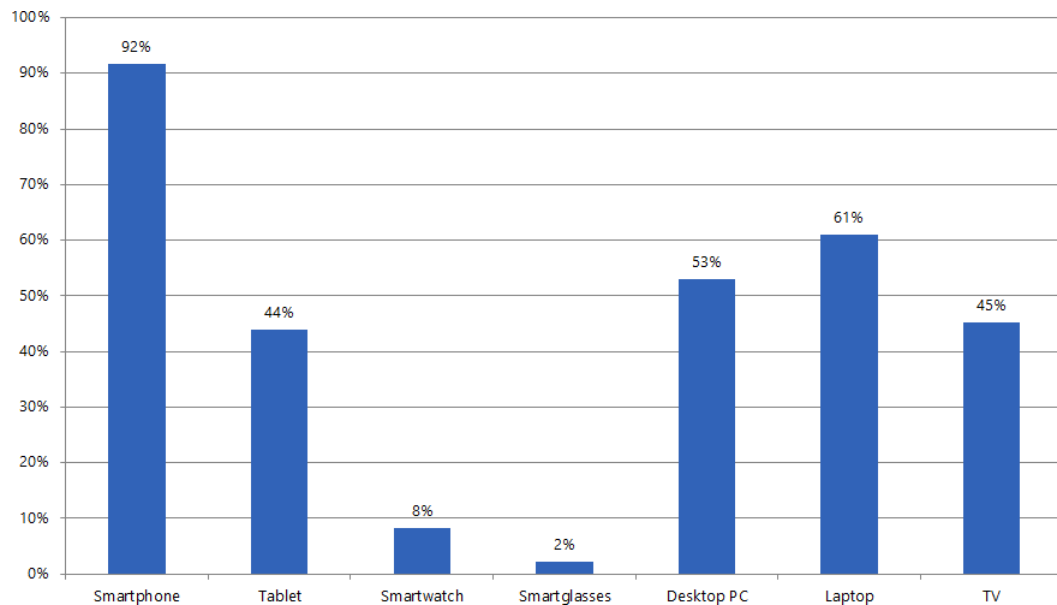


Figure 4.3.: The results of the device ownership survey show that almost all users own a smartphone. Tablet, desktop PC, laptop and TV are around the 50% mark. Smartwatches and smartglasses are not yet popular.

4.2.1. Design

Similar to the first survey this survey was shown to users of the *Desktop Notifications* app. However this time we displayed the survey on Android devices instead of the desktop. Again we created a survey notification on the device. This time the survey notification was triggered if a notification from another app was shown. We set a maximum of one survey notification every 22 hours and created the notification with the lowest priority possible. We did not use sound or vibration and no icon was shown in the notification area of the device. The user had to open the notification center to discover the notification. Upon tapping the survey notification a survey view was shown to the user (see Figure 4.4; the full sized screenshot is available in appendix A.2). In this survey view we told the user the name of the app which triggered the survey and the user was asked to assume that he owns all of the devices shown in the survey. Below this introduction we displayed seven statements, all starting with the text “I would like to see this notification on my ...” followed by the same device types as in the first survey (smartphone, tablet, smartwatch, smartglasses, desktop PC, laptop and TV). We asked the users to rate these statements on a Likert scale from 1 (*Strongly agree*) to 5 (*Strongly disagree*). Underneath the statements we added an optional text field for comments. Again we translated the survey into English and German and used the set language of the Android device to determine the language to be displayed, with the German version displayed on German devices and English on all others.

4. Device and Notification Studies

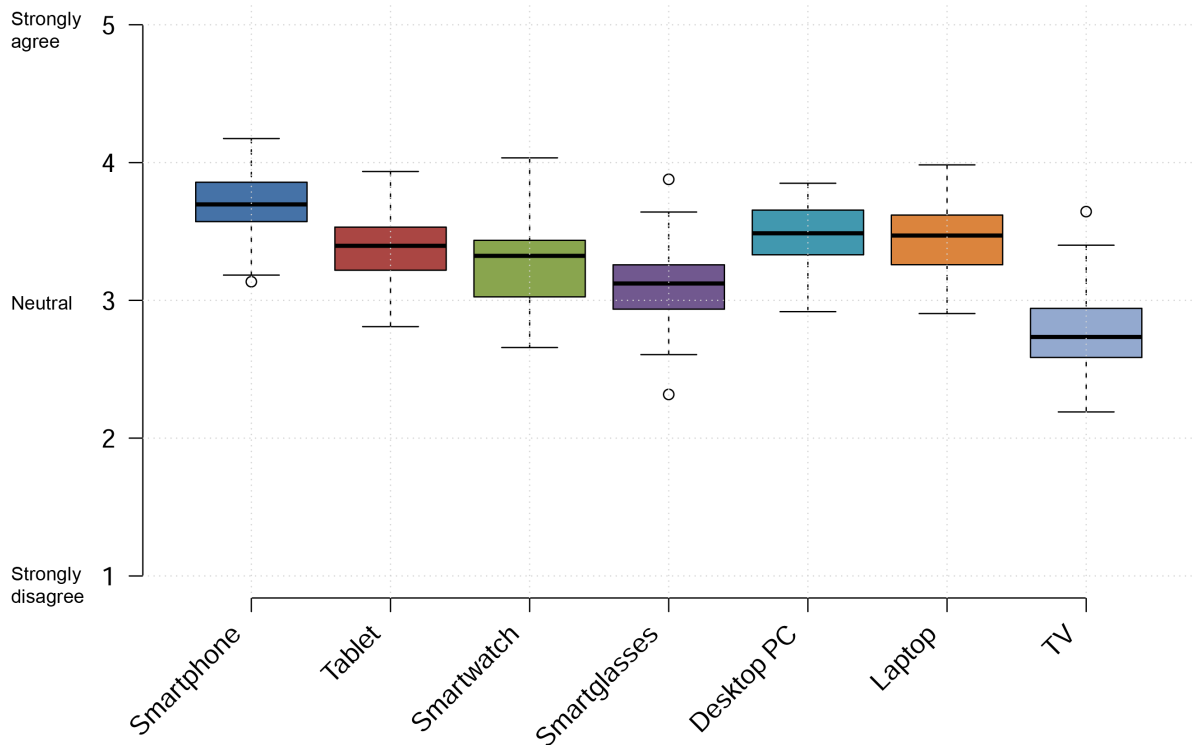


Figure 4.5.: Degree of agreement to the statements “I would like to see this notification on my <device type>” on a Likert scale from 1 (*strongly disagree*) to 5 (*strongly agree*). For the box plot the ratings of 46 apps were used, each with more than twenty participants.

When we looked at the number of votes per app we immediately noticed a long tail. Many apps were only rated by one user. For further analysis we cut off the long tail by only selecting apps with ratings from more than 20 users. This reduced the number of apps to 46 (available in appendix A.1). We aggregated the ratings for these apps and created the boxplots shown in figure 4.5. The boxplots show the agreement to the statement “I would like to see this notification on my <device type>”. The average rating for all device types except *TV* were rated *neutral* or higher. The highest rating was received by the *smartphone* category ($M = 3.69$, $SD = 0.22$), followed by *desktop PC* ($M = 3.46$, $SD = 0.24$) and *laptop* ($M = 3.44$, $SD = 0.23$). Similar to the first survey, these results are likely influenced by the fact that the survey was shown exclusively to users of the *Desktop Notifications* app, a group of users that already show interest in receiving notifications on the desktop. The next highest ratings were received by the categories *tablet* ($M = 3.88$, $SD = 0.28$), *smartwatches* ($M = 3.26$, $SD = 0.31$) and *smartglasses* ($M = 3.10$, $SD = 0.28$). The device type with the lowest rating was *TV* ($M = 2.77$, $SD = 0.27$).

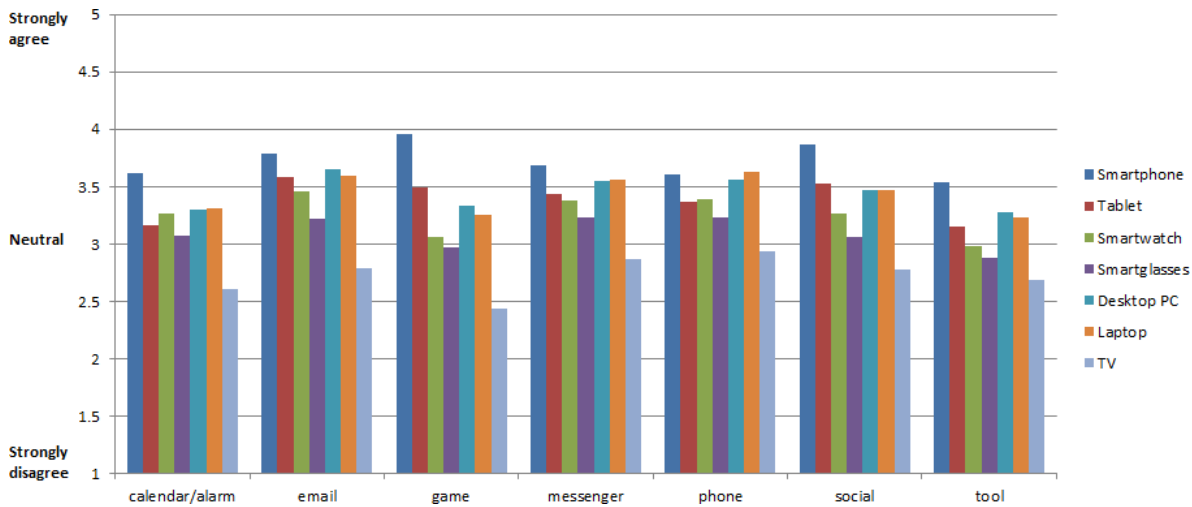


Figure 4.6.: The 46 most rated apps grouped into categories. The results of each category look similar to the combined results of Figure 4.5 with the exception of the *calendar/alarm* and *phone* categories, where the *smartwatch* ratings are higher than the *tablet* ratings.

To gain further insight into the data we categorized the 46 apps into seven categories: *calendar/alarm*, *email*, *game*, *messenger*, *phone*, *social* and *tool*. As shown in Figure 4.6 the distribution looks similar to the previously discussed boxplot, with the *smartphone*, *desktop PC* and *laptop* categories receiving high ratings. Noticeable exceptions are the slightly higher ratings of *calendar/alarm* and *phone* notifications on *smartwatches* compared to *tablets*. This is likely because these notifications often require an immediate reaction and thus are more useful on a device which is always worn by the user.

In this survey the optional comment field was hardly used. However we received negative feedback from multiple users who disliked our approach of creating notifications for the survey, even though they were silent. Because of this we later added a prominent opt-out link at the top of the survey.

4.3. Discussion

Our goal was to learn more about notifications on multiple devices. We conducted two studies targeted at users of the *Desktop Notifications* app. In the first survey we asked which kind of devices the users own. The high distribution of smartphones, desktop PCs and laptops were expected since those devices are required to be able to use the app. Tablets and TVs were common as well. However we did not ask for the exact models of the devices, so we did not find out which operating systems were installed on the tablets, nor whether or not the the

respective TV was a “smart” TV. In the second survey we asked the users on which kind of devices they prefer to receive notifications. The smartphone, desktop PC and laptop categories received a high rating, followed by tablet, smartwatch and smartglasses. TVs received the lowest rating.

Taking both surveys into account we decided to focus further work in this thesis on smartphones, tablets and desktop computers, because of their high ratings and high distribution. We determined that smartwatches and smartglasses are not yet widespread enough and TVs received a too low rating.

5. Implementation of the Framework

In this chapter we discuss the implementation of the prototype. First we describe the existing implementation, which was used as starting point. We then describe the changes to the *Desktop Notifications* [36] Android app as well as its server-side components and the Google Chrome extension.

5.1. Existing Implementation

The existing implementation [36] consists of an Android app, server-side components and a Chrome extension. The Android app was published in the Google Play Store¹ and is available for devices running Android 2.3.3 and later. The server-side components consist of PHP scripts and MySQL databases. The Google Chrome extension was published in the Chrome Web Store². On its first run, the Android app requests a unique code from the server. The code is randomly created and consists of 16 alphanumeric characters. The server adds the newly created code to a table in the database to ensure that every code is unique. The Android app stores the code persistently and displays it to the user. To connect the Android app with the Chrome extension the user has to enter the code provided by the Android app into the Chrome extension. When the Android app detects a new notification, it sends the notification, encoded in the JavaScript Object Notation (JSON) [8], to the server. The JSON object contains the content of the notification, meta data and the pairing code. A PHP script on the server processes the request and stores the data in the MySQL database. The Chrome extension periodically asks the server if notifications for a certain pairing code were added to the database. Depending on the amount of notifications a user receives, the extension polls every 10 to 60 seconds. The server selects all new entries in the database for the given code since the last request and returns the data as a JSON object. If the result is not empty, the Chrome extension creates a notification by calling Chrome's notification API. The package name of the Android app that created the notification on the Android device is used as an identifier for the notification on Chrome. New notifications from the same app will overwrite existing notifications.

¹Desktop Notifications in the Google Play Store <https://play.google.com/store/apps/details?id=org.hcilab.projects.notification>

²Desktop Notifications in the Chrome Web Store <https://chrome.google.com/webstore/detail/desktop-notifications-for-giicnncicnopjohcpamieklkiacdoeni>

5.2. Android App

In this section we discuss the changes to the *Desktop Notifications* Android app. For this thesis we changed the setup process, added the ability to receive and dismiss notifications, improved the settings and added a notification history.

5.2.1. Setup and Device Pairing

The existing version of the Android app, which we used as a starting point, consisted of two views: The main view and the settings view. As shown in Figure 5.1, the main view guides the user how to set up the app and pair the devices in the first three steps, with a fourth step to test the setup.

The first, second and fourth steps of the setup were left largely unchanged. The first step still asks the user to grant the app access to the notifications. The notification access is not covered by Android's permission system and has to be explicitly granted by the user in the system settings. If the user did not yet grant access a button is displayed which, depending on the Android version, will open the accessibility settings (Android 2.3 - 4.2) or the notification access settings (Android 4.3 and newer). In both cases the user is asked to click the button, check a checkbox in the system settings, confirm the action and then return to the app. The button is then replaced with the text *Enabled*.

The second step in the setup tells the user to download either the Chrome or Firefox browser extension. The user is shown short URLs leading to either the Chrome Web Store or the Firefox add-on marketplace. Because the links have to be opened in the desktop browser rather than the Android device, we intentionally left them unclickable.

In the third step previously a random pairing code generated by the server was shown to the user. While this works for a one-to-one device mapping, it doesn't work well in a multi-device environment because the user would have to pair every device with every other device. Another drawback of the code-based pairing system was that typos committed while entering the code caused the pairing to fail. Also the security aspect was rather lacking, because copying the code provided the ability to access the notifications. Thus we were looking for a pairing system that makes it possible to assign devices to a user in as few steps as possible and at the same time with improved security. We solved this challenge by implementing an authentication system based on OAuth 2.0 [22]. Because the Android app is distributed by the Google Play Store, which requires a Google account to download apps, we decided to use this account to authenticate the user. To do so we had to register the app in the Google developer console. This registration allows the user to grant the app access to his account. After access to the account has been granted, the Android app is able to generate an authentication token which is valid for approximately an hour. This token is sent along with the data for every server request. The server then validates the token and extracts the account ID of the user. This ID is unique

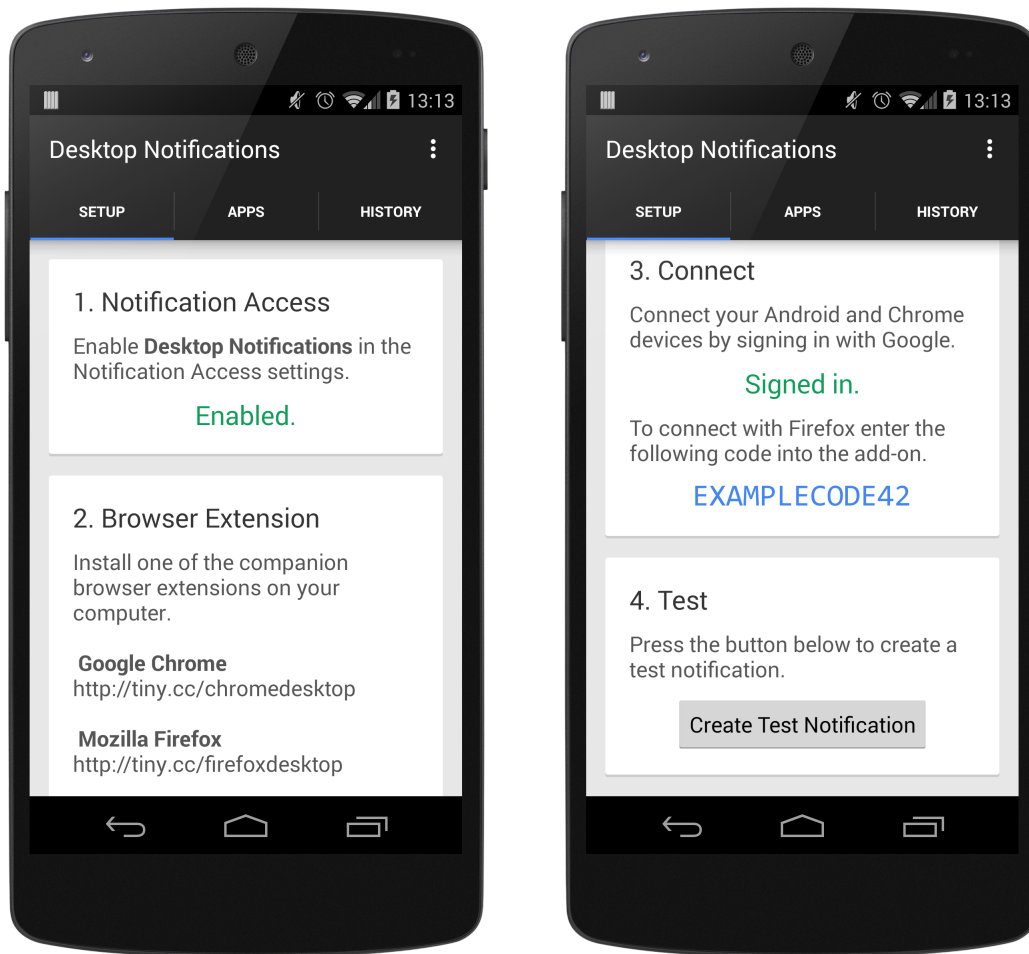


Figure 5.1.: The setup process of Android devices. — *Step 1:* Grant the app access to read the notifications. *Step 2:* Install browser extension. *Step 3:* Sign into the Google account. *Step 4:* Create a test notification.

for every account and does not change for the lifetime of the account. If the user signs in with the same account on every device, we can use this ID to pair the user's devices.

Figure 5.2 shows the sign-in flow from the user's perspective. The user is shown a *Sign in with Google* button. After tapping on the button the app looks for Google accounts on the device, and if multiple accounts are found an account picker is shown. On the very first sign-in a modal dialog is shown, asking the user if the app is allowed to access the account. This permission has to be granted only once and for every following sign-in the user is not asked again. It's also possible to register multiple apps in the Google developer console which all share this access. The account data which can be accessed by the app can also be controlled by so-called scopes. Since we only need the access to authenticate the user, we requested the most basic scope (*profile*) which enables us to access the unique account ID and public information. After

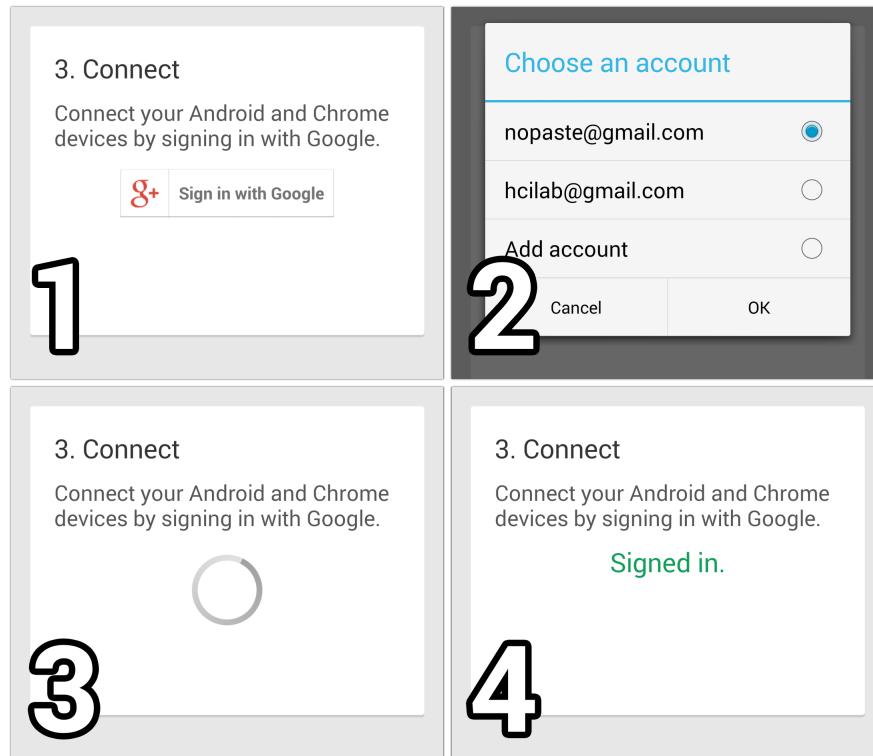


Figure 5.2.: The sign-in flow of the Android app. — (1) The user taps on the *Sign in with Google* button. (2) If there are multiple accounts on the device, the user chooses an account or adds a new account. (3) The user is authenticated and the device is registered on the server. (4) The user is successfully signed in.

the access is granted, the app generates an authentication token which is sent to the server to register the device. The server then validates the token by requesting the token validation endpoint provided by Google. If the token is valid, the endpoint returns the unique account ID of the user. The server then stores the device information along with the account ID in the database and tells the app that the device registration was successful. The sign-in button is then replaced by the text *Signed in*.

This device pairing system allows the user to pair a new device with only one tap on a button once access is granted, leaving little room for mistakes. Furthermore security is improved since an attacker now needs access to the Google account in order to intercept the notifications. Because OAuth is an open standard for authentication it is possible to add further authentication providers with little effort.

5.2.2. Receiving Notifications

In order to synchronize the notifications across the user's devices, the Android app has to receive events about new notifications from the server, similar to the Chrome extension. However, unlike the Chrome extension, an app on a mobile device should never continuously poll for new data in the background. The polling would cause the device to stay awake and quickly drain the battery. Sending push messages from a server to a mobile device is a common problem and because of this there are several services that try to solve it. For the Android platform Google offers the Google Cloud Messaging (GCM) service. The service runs in the background of Android devices and maintains a single connection to Google's servers. Third party apps can hook into the background service so even if there are multiple apps listening for push messages only one outbound connection is required. To use the GCM service we had to register the package name of the app and the IP address of our server in the Google developer console. In the app we had to implement event listeners for new push messages. In addition the app has to register itself with the GCM background service running on the Android device. In this registration process a unique key is generated which is used by the server to send messages to specific devices. The key is sent to the server in the authentication step we described in the previous section and it is stored together with the account ID in the database.

To send a message to a device, the server has to send the data and the key of the device to a GCM endpoint. The GCM service then takes care of delivering the message to the mobile device. The GCM background service on the mobile device receives the data and invokes the third party app. In our case we use these events to create synchronized notifications. As shown in Figure 5.3 a synchronized notification shows the title and content in addition to the name of the original device. This implementation behaves just like that of the Chrome extension. The notifications are grouped by the package names of the apps that created them. If a notification from a certain app already exists, it is updated. If the user clicks on the notification, the full text of the notification is displayed. These synchronized notifications can optionally play a notification sound or cause the receiving device to vibrate.

Using a push message system triggered by the server has the additional benefit of being faster than having the device poll every 10 to 60 seconds. Although the message has to be sent to the GCM endpoint first we observed an approximate transmission time of one second. For other platforms other services similar to the GCM push-service exist.

5.2.3. Dismissing Notifications

When synchronizing notifications across devices, we want to avoid that the user has to dismiss the same notifications on every device. Thus it is necessary to broadcast not only events about new notifications but also events about removing them. The API we use to listen for new notifications can also be used to listen for remove events. Since remove events only differ in

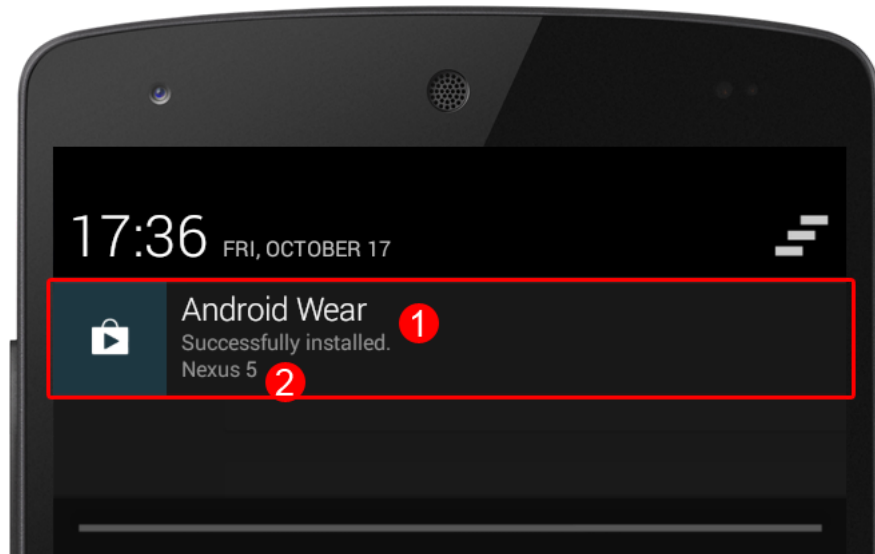


Figure 5.3.: Synchronized notifications show (1) the title and content of the original notification and (2) the name of the device on which the notification was created.

that they close a notification instead of opening it, we were able to use the existing architecture to deliver the events to the user's devices. As described in the previous section the data is first sent to the server, which forwards it to the GCM endpoint. The GCM service then takes care of delivering the data to the user's devices. On each device the Android app is then invoked and depending on whether the message contains a new notification or a remove event a notification is created or removed.

Using this system the user can dismiss a notification from any device, regardless on which device the notification was actually created first. Normally Android does only allow the removal of a notification if the app created it itself. However because the user granted our app notification access during the setup we can access methods that would otherwise not be available, like closing *any* active notification.

5.2.4. App Settings

To give the user control over which apps are allowed to send notifications we implemented the view shown on the left side of Figure 5.4. The view contains a list of all apps that have previously created a notification on the device. By tapping on the checkbox the user can enable or disable sending notifications for specific apps. By default all apps are set to send notifications. Below the name of the app a notification count tells the user how many notifications were created by the app. This information is useful to quickly detect apps that create a huge number of notifications.

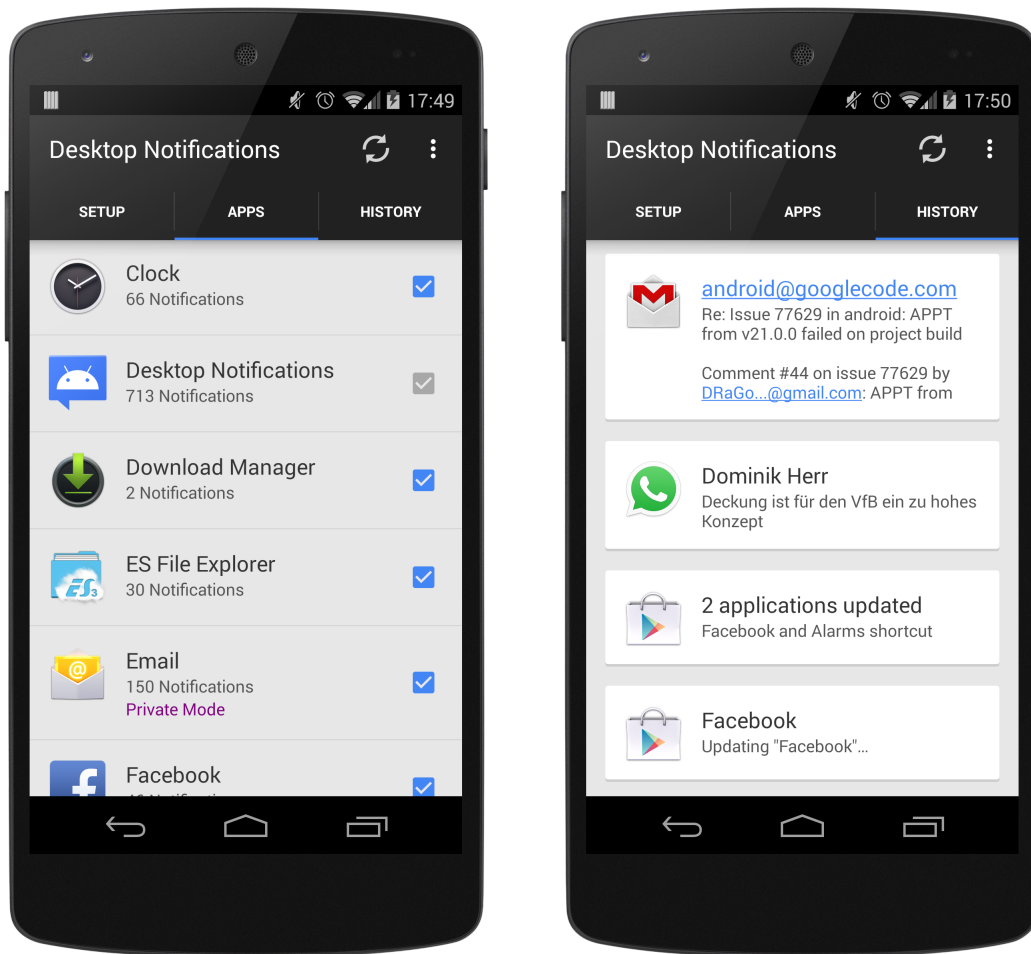


Figure 5.4.: *Left:* List of apps which are allowed to send notifications. “Private Mode” notifies but doesn’t send the text. *Right:* A history of all past notifications from this device.

This list of apps was already available in the existing version of the app, however it was shown at the bottom of the settings view and most users did not discover it. Thus we moved the list into a tab view so the users can reach it with one swipe from the setup view. The code related to the list had to be rewritten from scratch in order to move it from the settings view into the tab view. We used this opportunity to add a new functionality to the list. To preserve the user’s privacy the app already had the option to disable sending the content of notifications for all apps. We decided to change this global setting to a per app setting. The so called “Private Mode” can be enabled by long pressing on an app in the list. For existing users these changes reset all previous settings.

5.2.5. Notification History

Because it is now possible to dismiss notifications across devices we were worried about the user accidentally dismissing notifications. Thus we implemented the history view which displays all notifications that have been created on this device in chronological order. As shown on the right side in Figure 5.4 the view is accessible by clicking or swiping to the history tab. Each history entry contains the title and content of the notification in addition to the icon of the app which created the notification. The history is limited to a fixed number of notifications and can be cleared by the user. To preserve the user's privacy the history can be disabled at any time in the settings of the app.

5.2.6. Paired Devices and Settings

Since we now pair devices using the user's Google account it is possible to retrieve a list of all devices that are associated with the user's account. We implemented a view (see left side of Figure 5.5) that requests all paired devices from the server for a given account. The list shows the names of the devices ordered by the dates each device was added. The list can be accessed from the app's menu.

In the settings view (see right side of Figure 5.5) the user previously had the option to restrict sending notifications to Wi-Fi networks, disable sending the content of the notifications and disable sending for certain apps. As described in section 5.2.4, we moved the last two options to a more accessible view. The Wi-Fi setting remained and we added options to assign a name to the device and disable the notification history. The device name is used in synchronized notifications on other devices to declare the origin of the notification and in the list of connected devices. We generate a default device name based on the device's model name which is provided by Android. Because the model name is not always useful, we added a list of popular devices which maps the model name to a readable name. If the model is not in this list it uses the manufacturer's name instead. The device name can be freely edited by the user. Because the app is now able to receive notifications from other devices we added options related to incoming notifications. The user can enable a notification sound and control the vibration.

5.3. Server Architecture

To support multiple devices and the dismissal of notifications we had to rewrite the server-side components from scratch. The previous implementation was a one-way street: The Android app pushed data to the server and the browser extension used polling to retrieve the data from the server. However sending data from the browser to the Android app was not possible.

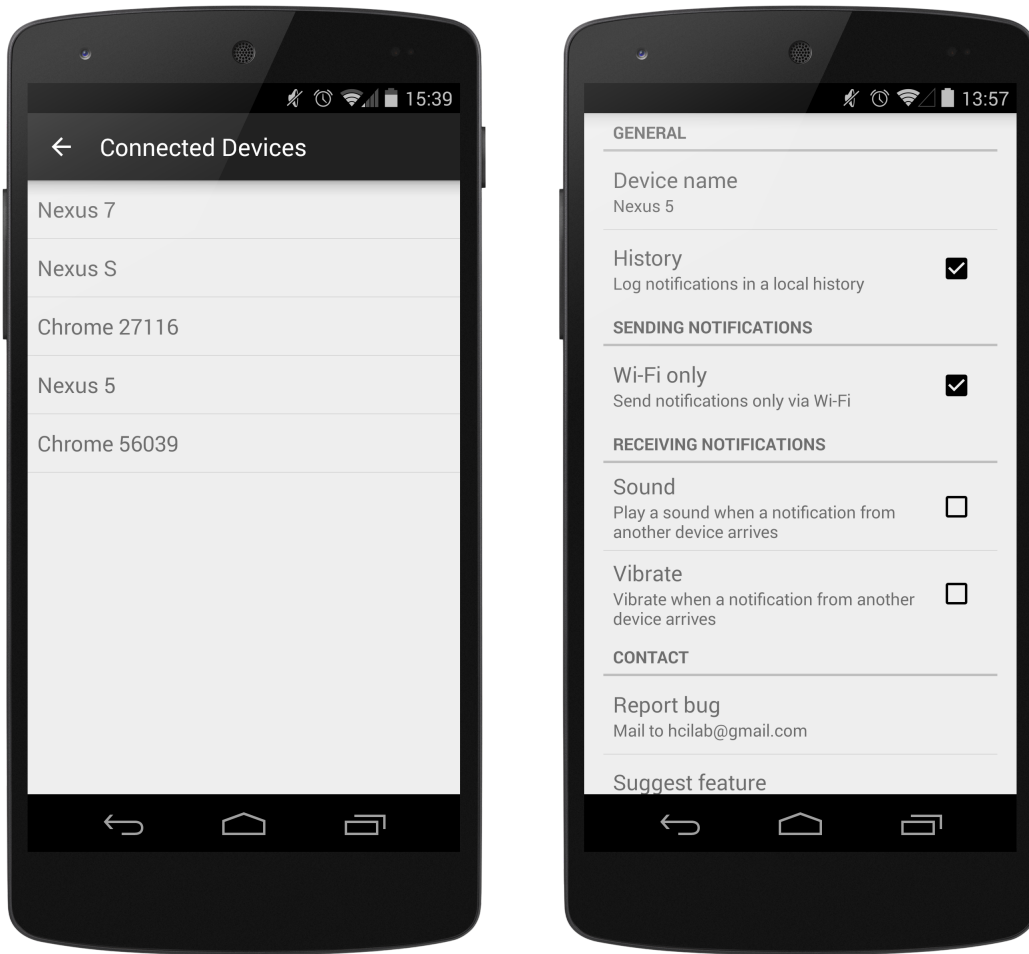


Figure 5.5.: *Left:* The list of devices associated with the user. *Right:* The settings screen allows the user to set a device name, enable the logging of notifications in a history and set notifications to be sent only through Wi-Fi. Synchronized notifications from other devices can be set to play a sound and/or vibrate the device.

5.3.1. Authentication

In the new version of the server every request has to be authenticated. Possible requests are: pairing a new device, retrieving a list of all paired devices and broadcasting events about new or removed notifications. To authenticate the user an authentication token is generated by the Android app or Chrome extension which is sent with every single request. The authentication token is valid for approximately one hour. Before the server can process the request itself it has to validate the token. This is done by sending the token to the Google token-info endpoint. The result contains information about the app that created the token, the Google account ID and the remaining token lifetime in seconds. In the case of an invalid token an error is returned

instead and the server will reject processing the request. Retrieving the information from the token-info endpoint can be expensive, especially when done dozens of times per second. Thus we cache the result for the remaining lifetime of the token in an in-memory database table. This ensures that during the lifetime of the token the lookup of the account ID is very fast for subsequent requests and requires no further network requests.

All requests from the Android app and Chrome extension to the server are sent via HTTPS and use a certificate provided by the German National Research and Education Network³. This ensures transport security of all data which is an essential part of the system's security. Without encryption an attacker would be able to read the authentication token and interact with the server on behalf of the user and thus gain access to the user's notifications.

5.3.2. Device Registration

When the user clicks on the *Sign in with Google* button in the Android app or the Chrome extension, the key for the Google Cloud Messaging (GCM) service, information about the device and an authentication token are sent to the server. The server authenticates the user and saves the GCM key together with the account ID into the database. This database table is the core element of the server, because it allows us to retrieve a list of all devices for a specific user. We also store additional meta data like the device type, so we can distinguish between Android smartphones, Android tablets and Chrome instances. With this information it is possible to broadcast notifications to all devices at the same time or only certain device types.

5.3.3. Event Broadcasting

To keep the user's notifications in sync across all devices, every new notification and every notification dismissal has to be sent to the server and then broadcasted to all devices. Figure 5.6 shows the necessary steps to broadcast an event. In step (1) an event about a new or removed notification is sent to the server. Every request includes an OAuth 2.0 authentication token. The server uses PHP scripts to process the requests. Subsequently, in step (2) the server sends the authentication token to the token-info endpoint which is provided by the OAuth authentication service provider. The authentication service provider validates the token and returns the user's account ID. The server uses the account ID to request all associated devices from the MySQL database. The result of the request is a list of device keys which can be used to send push messages to the devices via the Google Cloud Messaging (GCM) service. After that, in step (3), the server forwards the event to the GCM endpoint. In step (4) the GCM service pushes the event to the user's devices. The devices receive the event and, based on the contents of the event, create a new notification or remove an existing one.

³German National Research and Education Network <https://www.dfn.de/en/>

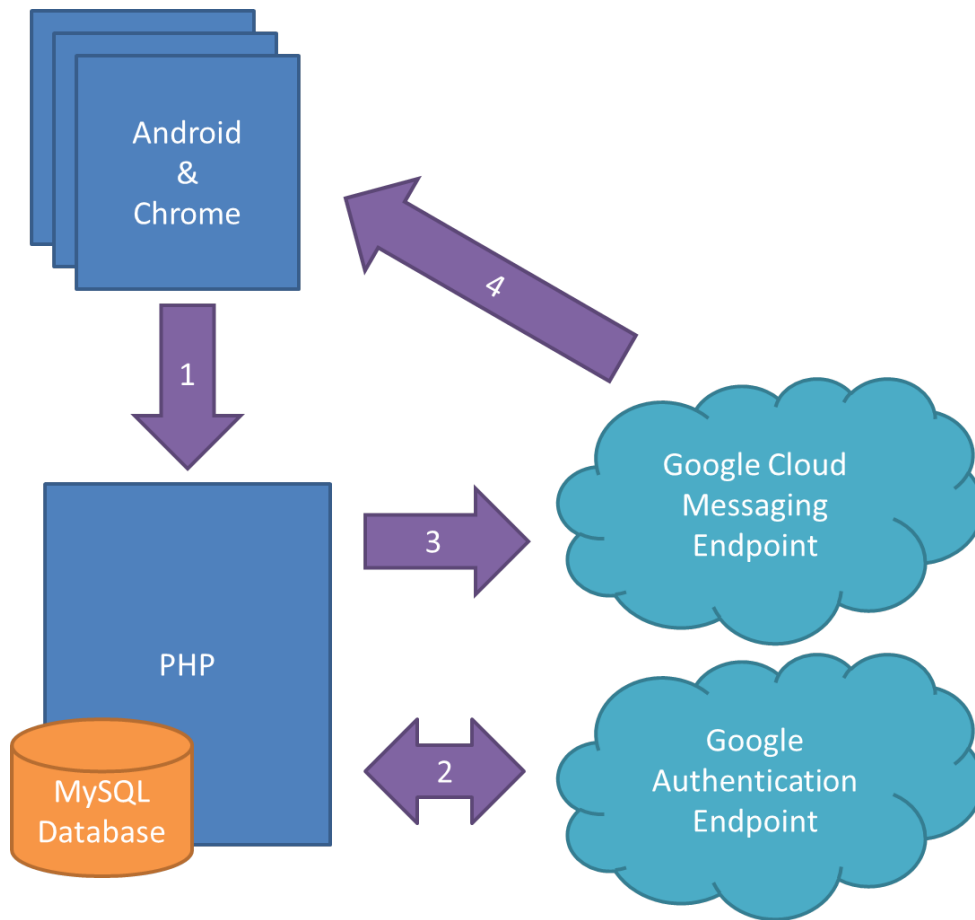


Figure 5.6.: The interaction between the Android app, Chrome extension, server-side components and Google services.

In listing 5.1 an exemplary JSON object for a new notification is shown. It consists of all necessary data that is required to display the notification on other devices. Because every request has to be authenticated, the JSON object contains an authentication token. Android notifications are identified using the the package name of the notifying app, a notification ID and an optional notification tag. Besides the package name the actual name of the app is also included. The title and content fields contain the content of the notification and are omitted if the user disabled sending the content for an app. The time field represents the Unix time stamp of the creation time in milliseconds. The device name is used to indicate the origin of the notification and the device type specifies the type of the device which triggered the event. Possible values for the device type are Android smartphone (*android_phone*), Android tablet (*android_tablet*) and Chrome browser (*chrome*).

For comparison, listing 5.2 shows an exemplary JSON object which is used to signal the dismissal of a notification. Again an authentication token is included in the request. In this

5. Implementation of the Framework

```
1 {
2     "access_token":    "ya29.UQC5W8Tjny ... kdN-uBKANL8",
3
4     "package":        "com.whatsapp",
5     "notification_id": 42,
6     "notification_tag": null,
7
8     "appname":        "WhatsApp",
9     "title":          "Dominik Weber",
10    "content":         "Example Notification",
11
12    "time":            1415630510793234,
13
14    "device_name":     "Nexus 5",
15    "device_type":     "android_phone"
16 }
```

Listing 5.1: Exemplary JSON object for creating a notification.

```
1 {
2     "access_token":    "ya29.TE9xms8PU ... P-roZoAb",
3
4     "package":        "com.whatsapp",
5     "notification_id": 42,
6     "notification_tag": null,
7
8     "time":            1415630514391345,
9
10    "device_type":     "chrome"
11 }
```

Listing 5.2: Exemplary JSON object for removing a notification.

example the remove event was not triggered on the same device as the creation event. Because of this the authentication token is different, however both tokens will return the same account ID. The package name, notification ID and notification tag match the values of the create event. The Android app and Chrome extension will compare the values and close all notifications with matching values. The name of the app, title, content and device name are omitted in the remove event, because they are not needed. The time field indicates the time of the dismissal and the difference between the time stamp in the create and remove events equals the duration for which the notification was shown. Similarly, the device type field contains the type of device which was used to dismiss the notification.

5.4. Google Chrome Extension

To support the new functionality of synchronizing notifications across devices we also updated the Chrome extension. Similar to the Android app we updated the way devices are paired and how notifications are received by the Chrome extension.

5.4.1. Setup and Device Pairing

The setup process of the Chrome extension previously consisted of entering the pairing code which was shown in the Android app. Because our goal was to assign all devices to a single account ID, we replaced the code-based mechanism with a sign-in button (as shown on the left side of Figure 5.7). The button allows the user to sign in with a single click if he previously granted the app access to his Google account. In the case that the permission to access the account was not granted yet, a pop-up window will open asking the user to do so. The permission to access the Google account is shared across the Android app and the Chrome extension, because we linked both projects in the Google developer console. After the user is successfully signed in, the text *Signed in* is shown along with options regarding the display duration of the desktop notifications and a checkbox to enable a notification sound.

5.4.2. Receiving Notifications

Previously the Chrome extension used polling to periodically ask the server for new notifications. On mobile devices we decided not to use this technique because of the negative impact on battery life. This doesn't apply to desktop PCs or even laptops. However, constantly polling for new notifications creates many unnecessary requests and does not scale well on the server-side. Furthermore, polling every 10 to 60 seconds means that in the worst case a notification might be delayed for 60 seconds. Thus we decided to also implement a server initiated push message system for the Chrome extension. Although the Google Cloud Messaging (GCM) system we use for the Android app is sometimes referred to as "GCM for Android" in the documentation, Google added support for it in a recent update of Chrome. Similar to the Android version of the service, extensions can request a unique key for a particular Chrome instance. The key is sent to the server during setup and is stored in the database together with the account ID. The extension then only has to register an event handler to receive push messages from the server.

We also updated the extension to show the name of the origin device in desktop notifications. Similar to the Android app, the name is shown below the content of the associated notification and enables the user to see the origin at a glance.

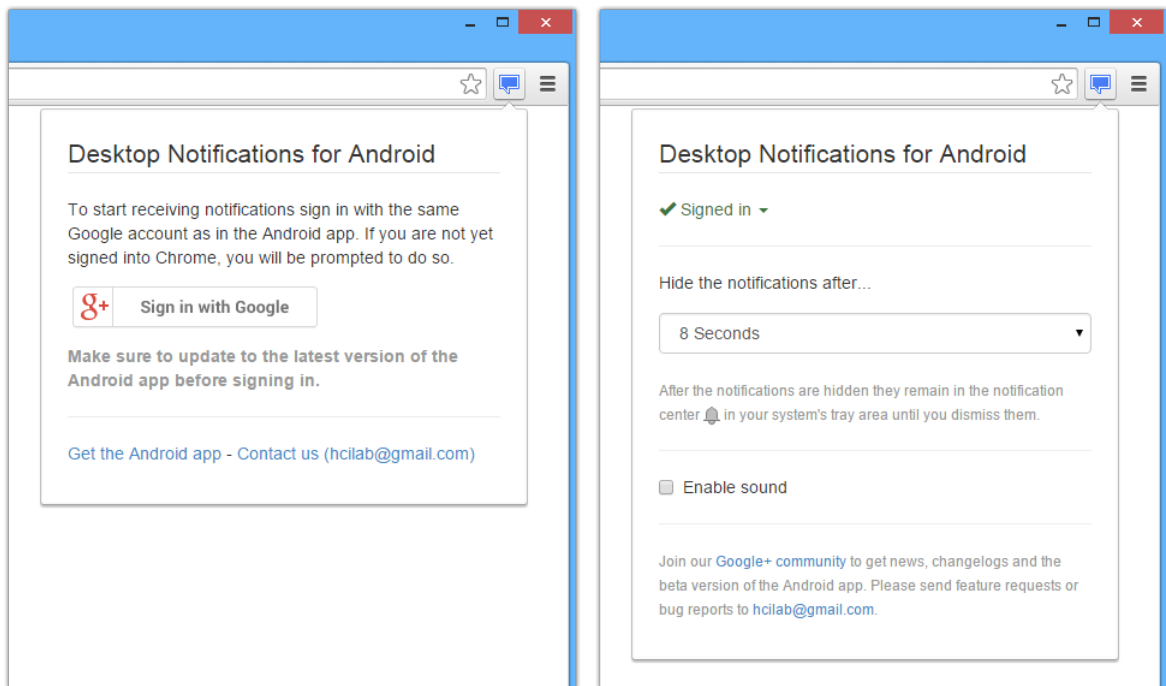


Figure 5.7.: The setup and settings dialogs of the *Desktop Notifications* extension for Google Chrome. — *Left:* The user is instructed to sign into the same Google account as on the Android device. *Right:* Confirmation of the successful sign-in and settings to change the duration of desktop notifications and enabling a notification sound.

5.4.3. Dismissing Notifications

Desktop notifications are only shown for a couple of seconds and are then moved into Chrome's notification center, which behaves similarly to Android's notification center. The user can review the notifications and dismiss a single notification or all notifications at once. Before, dismissing a notification had no other effect than removing it from the user interface. We added an event listener to the notifications which notifies the extension about dismissed notifications. The extension then forwards the event to the server, which broadcasts it to all of the user's devices.

At the same time the extension can now receive remove events from the server and close notifications in Chrome's notification center. By broadcasting events about new and dismissed notifications on every platform we are able to keep the user's notifications on all devices in sync.

5.5. Extensibility

Currently the system only supports Android-based devices and Google Chrome on the desktop. However we designed the architecture in a loosely coupled manner which can be easily extended to support other connected devices. For example, the Google Cloud Messaging service can only push messages to Android-based devices or Chrome instances. However other push messaging service providers exist. Apple offers the Apple Push Notification (APN) service⁴ to target iOS-based mobile devices and Microsoft offers Windows Push Notification Services (WNS)⁵ for Windows and Windows Phone apps based on the Windows Runtime. To support other browser extensions besides Chrome, for example the Firefox add-on, WebSockets [18] could be used. WebSockets are low-overhead persistent connections which enable bi-directional sending of messages between the browser and a server.

Wearable devices, like smartwatches, that do not have a direct network connection but rather use Bluetooth to communicate with a mobile device can, in some cases, also be integrated into the system. The requirement is that the wearable device synchronizes the notifications with the mobile device through Bluetooth. In one of our tests we used an Android-Wear-based smartwatch and paired it with an Android smartphone. The smartphone then was paired with an Android tablet using the *Desktop Notifications* Android app. A notification on the tablet would then show up on the smartphone and on the smartwatch. Dismissing the notification on the smartwatch would remove it from the smartphone and tablet. However in this case it's not possible to detect if the notification was removed on the smartwatch or on the smartphone.

In the current implementation the authentication mechanism is based on the user's Google account. Because Google implements the OAuth authentication scheme it is possible to use another authentication provider with little effort or give the user the choice to use one of multiple authentication providers. Most social networks can act as OAuth authentication service providers.

5.6. Limitations

A special case of Android notifications are so-called ongoing notifications. They notify the users about ongoing processes like download progress, the duration of ongoing phone calls or the currently played song. Typically these notifications are updated often. For example the duration of a phone call will update once a second to increase the timer. Updates to Android notifications are internally treated as removing the existing notification and creating a new one.

⁴Apple Push Notification (APN) service <https://developer.apple.com/library/ios/documentation/NetworkingInternet/Conceptual/RemoteNotificationsPG/Chapters/ApplePushService.html>

⁵Windows Push Notification Services (WNS) <http://msdn.microsoft.com/en-us/library/windows/apps/hh913756.aspx>

The notification API allows app developers to set a special flag to indicate that the notification is ongoing. Because of the huge amount of events created by this kind of notifications we filter them out using the “ongoing” flag.

To reduce the bandwidth that is required to broadcast notifications to all devices we decided to not send any images with them. On Android notifications show an icon of the app next to the text of the notification and in some cases, like text messages, the icon is replaced by an image of the contact. As a workaround we determined the top hundred most used apps and crawled the Google Play Store to download their default app icons. We bundled these icons with the Chrome extension and the Android app. For less popular apps a generic default icon is shown instead. We also do not transfer any sounds or vibration patterns.

Most Android notifications are actionable and tapping on them will, for example, open the related app. Because we don’t include possible actions when broadcasting the notifications to all devices, the synchronized notifications are not actionable. This can be confusing at times, for example, after creating a screenshot on an Android device a notification with the text “Click here to view the screenshot” is created, which would be visible in synchronized notifications as well. This also means that accepting calls or replying to text messages by tapping on a notification is not possible, something the users are used to doing.

The add-on for Mozilla Firefox was developed to reach more users and increase the number of installations of the Android app. Functionality-wise the Firefox add-on worked exactly the same as the Chrome extension. However we decided against updating the add-on to the new system, because updates for Firefox add-ons are manually reviewed by Mozilla before they can be rolled out to all users. The waiting time for the review process is usually several weeks and thus out of the scope of this thesis. Considering the fact that most users use the Chrome extension we decided to focus our work on the Chrome extension.

All devices require an active Internet connection for the framework to work. If any device is offline, it will be unable to send or receive events and the device will go out of sync.

5.7. Summary

We introduced a framework which makes it possible to synchronize the user’s notifications across Android smartphones, Android tablets and desktop computers. The user’s devices are paired by signing into the Android app or Chrome extension with a Google account, which, after granting the permission once, is done with only one tap or click. The user’s notifications are sent to a central server which broadcasts the notifications to all connected devices. Dismissing a synchronized notification on one device automatically removes it on all other devices. The user can disable sending notifications for certain apps or enable “Private Mode”, which will still notify the user but does not include the actual content of the notification.

6. Deployment and Usage Studies

In this chapter we describe the process of publishing the updates to the Android app and Chrome extension to the existing user base. We then analyze the state of the user base, their devices, and apps settings after two months. Furthermore we conducted a study to learn more about the usefulness of synchronized notifications across devices. We also updated the app once again to offer more options and looked into the resulting effects.

6.1. Publishing

We published the updates to the *Desktop Notifications* [36] Android app and Chrome extension in the Google Play Store and the Chrome Web Store. Updating both, the Android app and the extension, at the same time turned out to be a challenge for a number of reasons. For one, the update mechanisms of the Google Play Store and the Chrome Web Store behave differently. Updates to a Chrome extension are rolled out to all users automatically, without any user interaction, within several hours. The Google Play Store in comparison rolls out the updates rather slowly, over several days. Furthermore, not all users receive the updates automatically, because the Play Store allows users to disable automatic updates. If an app requests additional permissions the update has to be performed manually by the user, regardless of set update preferences. This was the case for the update to the Android app, as it requested new permissions for the implementation of the Google sign-in. When we released the update to the Play Store approximately 35,000 users had the current version of the app installed and approximately 12,000 users used a version that was no longer supported. In addition to the requirement of a manual update the replacement of code-based pairing with account-based pairing meant that existing users had to pair their devices anew.

We approached this challenge by temporarily supporting both the old and new versions of the Android app, Chrome extension and server-side components. Furthermore, we rolled out the updates in three phases:

Phase 1 In the first phase we published the updated Android app in the Play Store which triggered the manual update for all users. We explained the new pairing mechanism in the “what’s new” section of the Play Store. Because we didn’t expect all users to sign into the app right away, we modified the server to handle the notifications with both the old polling-based mechanism and the new push-based mechanism. We also updated

the Chrome extension to show a hint that the new pairing mechanism is now available. We monitored the user behavior over the course of several days and made sure that the update worked as intended.

Phase 2 In the second phase we updated the Chrome extension again to display the sign-in button to new users, while removing the option to use the code-based pairing mechanism. For existing users we added a notice that code-based pairing would be disabled soon.

Phase 3 Eventually half of the active Android app users updated to the new version. We then disabled the old polling-based mechanism on the server side and updated the Chrome extension once again to remove the code-based system completely. Existing users that had not paired their devices using the new account-based pairing system yet now saw a message explaining the update in detail.

This process happened over the course of two months. According to the Google Play Store statistics, approximately 8,000 users did not update the Android app. However, during the same time the app gained several thousand new users.

The following statistics describe the state of the Android app, Chrome extension and Firefox add-on at the time of writing in December 2014. One has to keep mind that the app and browser extensions were first published in December 2012 and updated for this thesis in October 2014. The Android app has been downloaded 298,841 times since its release and is currently installed on 58,265 devices. Of these active installations 35,863 (61.55%) use the newest version of the app and 22,402 (38.45%) use an older version which we disabled on the server side. The three most used devices are the Google Nexus 5 (5.81%), the Samsung Galaxy S3 (4.14%) and the Google Nexus 4 (4.07%). According to the Play Store the app is installed on 3,386 tablets with a display size ranging from 7 to just under 10 inches and on 2,598 tablets with a display size of 10 inches and larger. Most downloads originated from the United States (17.94%), followed by Germany (12.08%), India (6.31%) and the United Kingdom (4.86%). The app has been rated 8,155 times, with an average rating of 4.26 out of 5 stars and received over 1,700 comments.

The Chrome Web Store reports approximately 80,000 weekly users. The discrepancy regarding user count between the Chrome extension and the Android app indicates, that many users might install the Chrome extension out of curiosity but fail to install the Android app, or simply do not bother to remove the extension from Chrome after they decide to no longer use the app. The Chrome extension has been rated 616 times, with an average rating of 4.25 out of 5 stars.

The statistics dashboard for the Firefox add-on reports 67,814 total downloads and between 8,000 and 9,000 daily active users. It has been rated 16 times, with an average rating of 4.5 out of 5 stars.

6.2. Users and Devices

After two months we created a snapshot of the database and analyzed the list of devices which were registered with the service at the time. In total 33,888 users used their Google account to sign in and 53,565 devices were active. The number of active devices included of 27,933 (52.15%) Android smartphones, 2,214 (4.13%) Android tablets and 23,418 (43.72%) Chrome instances installed on desktop PCs or laptops.

We looked at the distribution of these three device types among the users. For example, if a user installed the Android app on two smartphones and the Chrome extension on three computers, we counted this as two distinct device types. Of the 33,888 users 19,004 (56.08%) have only one device type associated with their account, 14,409 (42.52%) users have two and the remaining 475 (1.40%) users all three.

The group of users that had only one device type associated with the app is interesting to look at, considering that at first glance they would not be able to use the app at all. However, because we only counted the distinct device types, a user could use the app to synchronize notifications between multiple phones or multiple tablets. Another possibility is that users used the Firefox add-on, which wasn't counted in these statistics, but still signed into the Android app with their Google account. A third possible group are users which started the pairing process but did not bother adding other devices. Breaking down the users with only one device type ($N = 19,004$) revealed that 11,497 (60.50%) users registered an Android smartphone, 1,082 (5.69%) an Android tablet and 6,425 (33.81%) a Chrome instance.

The 14,409 users which registered two distinct device types can be categorized into the combinations of Android smartphone with tablet (238 users, 1.65%), Android smartphone with Chrome (13,710 users, 95.15%) and Android tablet with Chrome (461 users, 3.20%).

We also logged the underlying operating system of the Chrome installations. Out of 23,418 Chrome instances 20,048 (85.61%) were installed on a computer with Microsoft Windows, 1,634 (6.98%) on OS X, 1,151 (4.92%) on a Linux distribution and 585 (2.50%) on Chrome OS. (All distributions are available as pie charts in appendix A.5.)

6.3. Private and Disabled Apps

Once a week, *Desktop Notifications* Android app automatically sends the list of apps that created at least one notification on the user's device to the server. The list includes information about whether or not the user has disabled sending notifications for certain apps and if the user enabled the "Private Mode" (notify, but don't include the content). With the update to the Android app we reset all previous settings, so the following collected data is the result of the app being used in the wild over the span of two months.

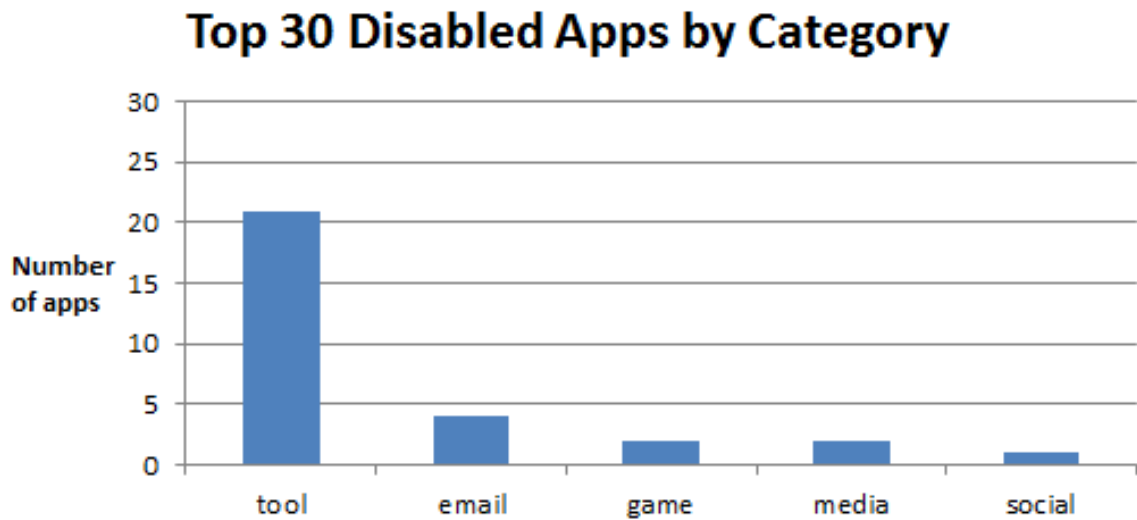


Figure 6.1.: Categories of the 30 most frequently disabled apps with at least one thousand users.

In total we saw that 36,249 different apps created at least one notification. Of these apps 6,051 had notifications disabled and 452 were marked as private by at least one user. We selected all apps that were used by at least one thousand users and calculated for each of them how many users disabled sending notifications for the app and how many enabled the “Private Mode”.

A total of 146 apps were disabled by at least one thousand users. We sorted the list of apps by the percentage of users that disabled notifications for them. From this sorted list we took the 30 highest percentage entries and categorized them. (The list and the corresponding categories are available in appendix A.3.) In Figure 6.1 we visualized the distribution of the categories in a bar chart. With 21 out of 30, most apps are in the *tool* category by a large margin. The other apps are divided into the categories *email* (3), *game* (2), *media* (2) and *social* (1).

In the same manner we extracted the list of apps that were set in the “Private Mode” by at least one thousand users. This list contains 113 apps. Again, we calculated the percentage, sorted the list and looked at the 30 most frequent apps. (This list of apps with the corresponding categories are available in appendix A.4.) As shown in Figure 6.2, *messenger* (16) and *email* (7) apps were set to “Private Mode” most often. The content of the notifications created by these apps often is of a private or confidential nature, yet at the same time receiving notifications from these types of apps is important to the users [36]. Not sending the actual content of the notification but still getting notified is a compromise many users seem to agree with. The same goes for apps from the *social* (2) and *calendar/alarm* (1) categories. With four apps the *tool* category is positioned at rank three. They consist of two anti-malware apps, an alternative app store for Android and an app related to online banking.

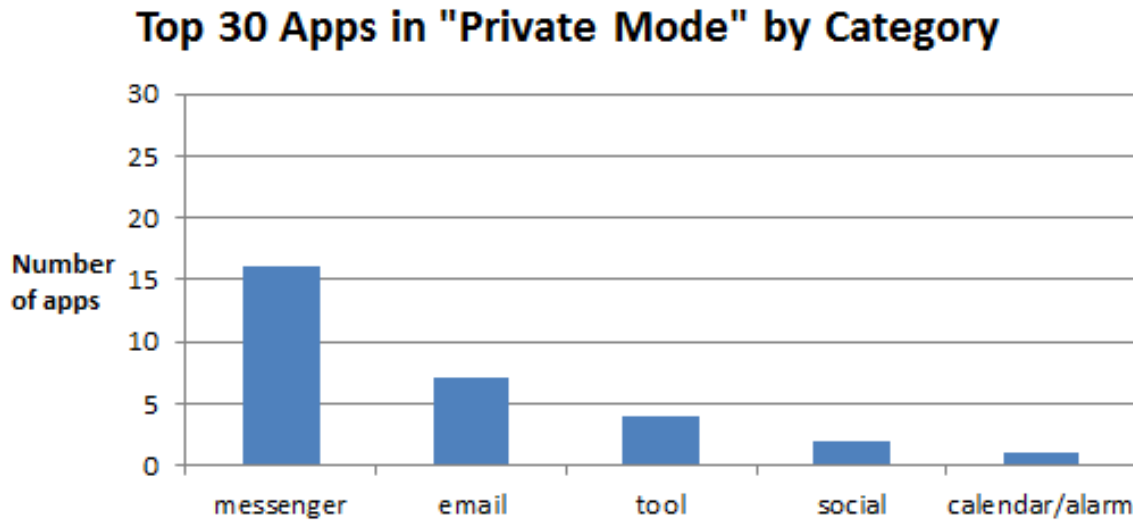


Figure 6.2.: Categories of the 30 apps that were set to “Private Mode” most frequently and which are used by at least one thousand users.

6.4. Usefulness of Synchronized Notifications

To learn more about synchronized notifications across different devices we asked the users of the app to rate their usefulness.

6.4.1. Design

We updated the Android app and Chrome extension once again and added a feedback button below some synchronized notifications. We limited the feedback button to the 46 apps which we already used in section 4.2 (see appendix A.1). Figure 6.3 shows the button below a synchronized notification on an Android device. Tapping on the *feedback* button opens a new view where the user is asked if they felt receiving this notification on the current device in addition to the origin device was useful to them. Both the name of the app origin device were explicitly named. The rating was done on a Likert scale from 1 (*strongly disagree*) to 5 (*strongly agree*). Tapping on one of the options immediately closed the view and sent the rating to the server. This feedback procedure was utilized on both Android smartphones and tablets. As seen in Figure 6.4, we implemented the same feedback form in the Chrome extension. We translated both forms to English and German and used the device’s set language to determine which language should be displayed. As usual, for German users we displayed the German translation and in all other cases the English version.

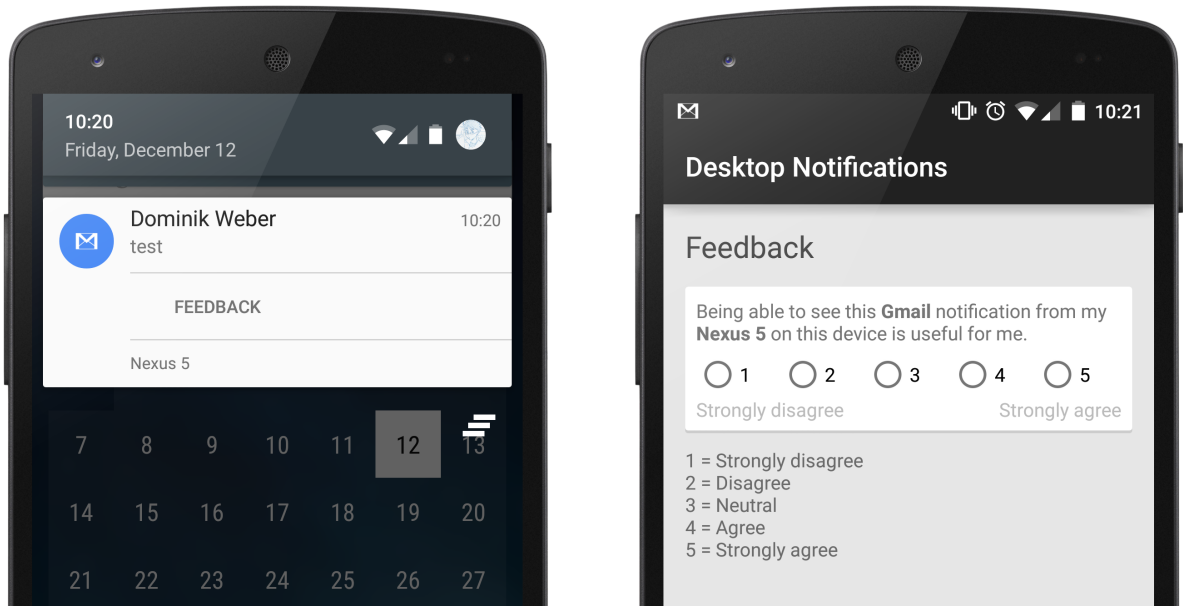


Figure 6.3.: *Left:* Feedback button below a synchronized notification on an Android device. *Right:* Feedback view shown after tapping on the feedback button.

6.4.2. Results

The feedback button below synchronized notifications was shown for two months. During this time 286 users rated a total of 609 synchronized notifications. 37.76% of the participants used a device set to English, 21.68% German and 8.04% Spanish. The complete language distribution is available in appendix A.5.

The origin of the synchronized notifications could either be an Android smartphone or tablet. With a smartphone as origin, the participants rated 43 synchronized notifications on another smartphones, 97 on a tablet and 107 on the desktop. Likewise, with an tablet as the origin, 71 notifications were rated on an Android smartphone, 1 on a tablet and 13 on the desktop.

For further analysis we categorized the apps again. The categories *calendar/alarm*, *phone* and *game* did not receive a meaningful amount of ratings, so we were only able to analyze the categories *email*, *messenger*, *social* and *tool*. Figure 6.5 shows the rating of synchronized notifications grouped by category and the device type which was used to rate the notifications. Notifications on the desktop received the highest rating in all categories. The *messenger* category ($M = 4.54$, $SD = 0.97$) is followed by *email* ($M = 4.12$, $SD = 1.41$), *social* ($M = 3.56$, $SD = 1.83$) and *tool* ($M = 3.04$, $SD = 1.52$). On smartphones the *messenger* category also received the highest ratings ($M = 3.97$, $SD = 1.59$), followed by *social* ($M = 3.04$, $SD = 1.7$), *tool* ($M = 2.74$, $SD = 1.58$) and *email* ($M = 2.24$, $SD = 1.6$). On tablets the *messenger* category again received the highest ratings ($M = 4.41$, $SD = 1.08$), followed by *email* ($M = 2.78$, $SD = 1.62$), *tool* ($M = 2.4$, $SD = 1.59$) and *social* ($M = 2.25$, $SD = 1.48$).

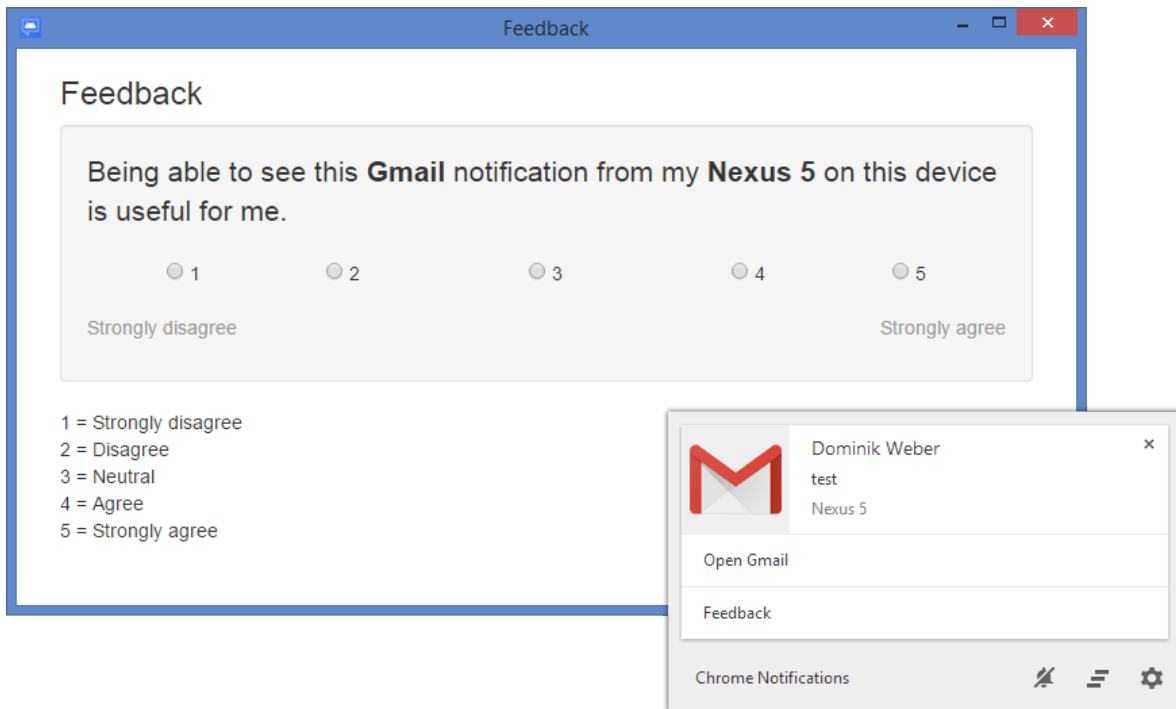


Figure 6.4.: *Bottom right:* Feedback button below a synchronized notification created by Chrome. *Top left:* Feedback window shown after clicking on the feedback button.

Overall the notifications from the *messenger* category received the highest ratings on all devices, verifying the statement “notifications are for messages” [36]. The high rating of *email* and *social* notifications on the desktop compared to smartphones and tablets is interesting. We assume that many users use browser-based email clients and social networks. As we explained in section 2.4.3 the Web Notifications API is partially supported but not yet widely used in the wild. This means the users have to rely on notifications from their mobile phones or tablets to receive notifications on the desktop. On the other hand email accounts and social networking apps might already be set up on all mobile devices resulting in duplicate notifications and thus a lower rating. It is also possible that the results were influenced by the name of the app (*Desktop Notifications*).

With the exception of the *messenger* category the results of this study are in stark contrast to the results of the study conducted in section 4.2. When asked on which kind of devices the users would like to see the notifications every device type except the TV was rated neutral (3) or higher. In this study only the desktop received comparable ratings.

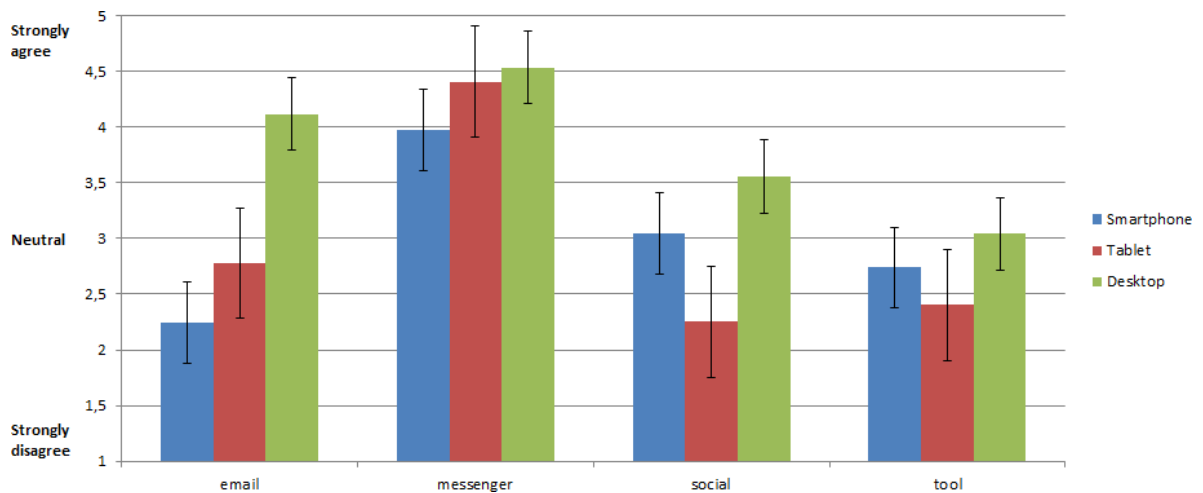


Figure 6.5.: Agreement to the statement “Being able to see this <app name> notification from my <device name> on this device is useful for me.” on a Likert scale from 1 (*strongly disagree*) to 5 (*strongly agree*). The ratings are grouped by the categories of the apps and the type of the device which was used to rate the notification. The error bars show the standard error.

6.5. App List Update

Because of the differences in the usefulness of synchronized notifications across device types we decided to update the Android app to support disabling apps from sending notifications to certain device types instead of disabling it completely. As shown in Figure 6.6, we replaced the checkbox next to every app with three icons. The icons include the characters *D* for desktop, *S* for smartphone and *T* for tablet. The current settings regarding each device type are shown through colors. A gray icon is shown if notifications from a certain app should not be sent to this device type. A green icon is shown if notifications should be sent, which is the default setting. When the user taps on an app a settings dialog is shown which allows the user to enable or disable sending notifications for this app to specific device types. We also moved the option to enable the “Private Mode” from the long-press menu to this settings dialog.

These changes mean that now all notifications are sent to the server, unless the user has disabled the sending of them for all device types. We extended the JSON messages by including the user’s settings. Furthermore we modified the server to filter by disabled device types instead of broadcasting the notifications to all devices. We published the update to all users. In the progress we reset the previous settings, so every user had all device types enabled for all apps after the update.

After three weeks we analyzed the settings. In total sending notifications for 2,540 apps was disabled by at least one user. For further investigation we only considered apps that had

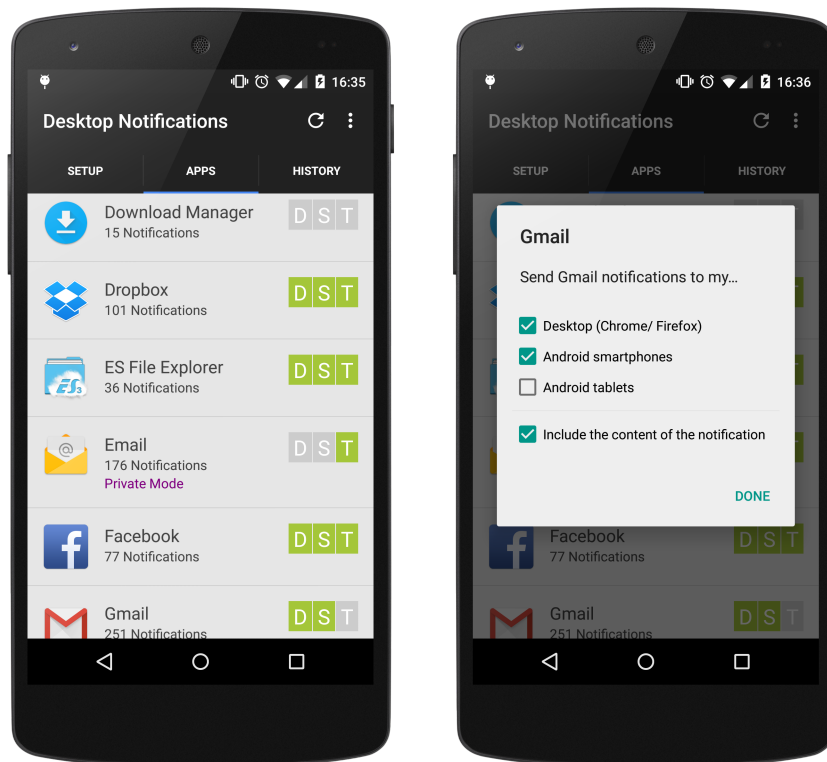


Figure 6.6.: Updated list of apps which are allowed to send notifications. — *Left:* The checkboxes next to every app have been replaced by three icons. *Right:* Tapping on an app opens a settings dialog.

notification sending disabled by at least one thousand users ($N = 127$) and sorted the list of apps depending on the percentage of how many users disabled notification sending for it. Figure 6.7 shows the 30 apps that had notifications disabled most frequently by category, divided into target device types. (The lists with their corresponding categories are available in appendix A.6, A.7 and A.8.) Similar to the findings of the previous analysis in section 6.3, most apps belong to the *tool* category. Notifications from apps of this category have been disabled from being sent to smartphones, tablets and the desktop equally, with 19 apps each. The *media* category comes second with 5 apps excepted from sending to smartphones and tablets and 6 apps to the desktop. For the *email* category the incidence of disabled notification sending to the smartphone and tablet is 3, with 4 to the desktop. The *calendar/alarm* category saw 3 apps excepted on both the smartphone and tablet, but none on the desktop. Only one app from the *game* category has been disabled from sending notifications to the desktop frequently.

Compared to the findings in section 6.3 most apps that had notification sending disabled are again from the *tool* category by a large margin. In both findings the categories *email*, *game* and *media* make an appearance but instead of the *social* category the *calendar/alarms* category appeared.

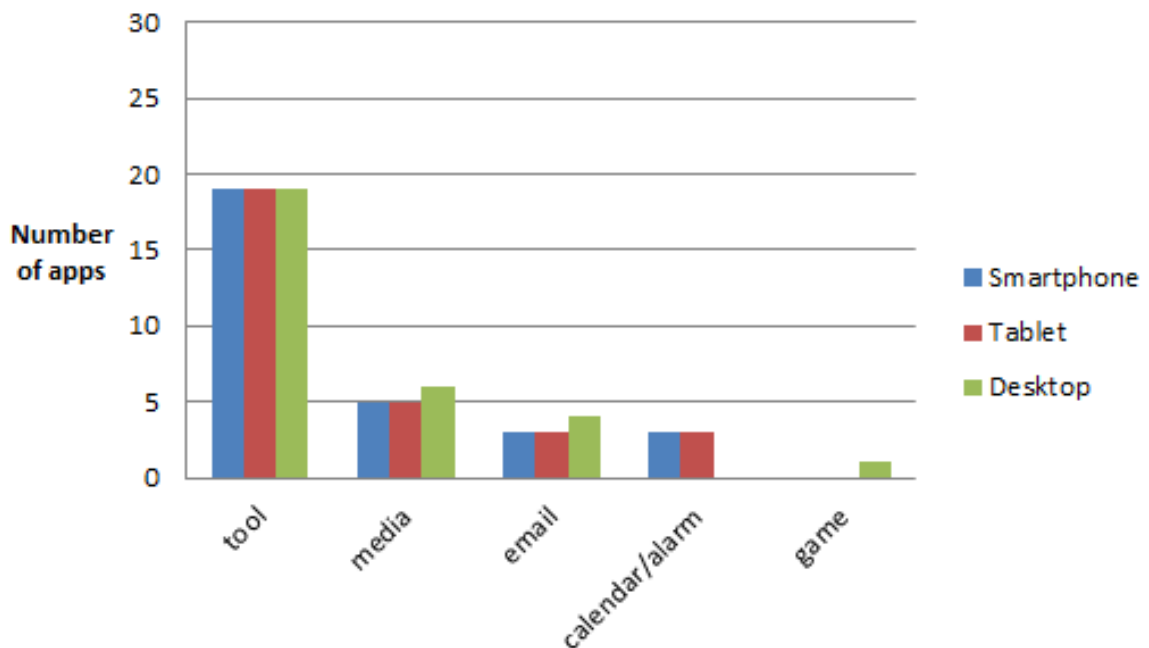


Figure 6.7.: Categories of the 30 apps that have been disabled from sending notifications most frequently, grouped by target device type.

6.6. Summary

In this chapter we described the process of publishing the updates to the Android app and Chrome extension to the existing user base. Because the users had to manually confirm the update of the Android app and had to pair their devices again, we performed this update in three phases. After most users adapted to the updates, we analyzed the user base, device distribution and which apps were commonly disabled or set into the “Private Mode”. The results showed that the users disabled apps from the *tool* category the most. The “Private Mode” was often enabled for apps related to communication. We then conducted a study about the usefulness of synchronized notifications. Notifications from the *messenger* category received high results on every type of device. The other categories showed mixed results, with the desktop receiving the highest rating compared to smartphones and tablets. We assume that the users tended to receive duplicate notifications thus lowering the usefulness of the synchronized notifications. Another explanation is that the results were influenced by the name of the app (*Desktop Notifications*). On the basis of these mixed results we updated the Android app to allow disabling the sending of notifications to certain device types instead of disabling them completely. We saw few differences compared to the previous analyses, with the *tool* category still having notifications disabled most often.

7. Conclusions and Future Work

In this chapter we recap the work of this thesis and draw conclusions from our findings. Finally, we suggest ideas for future work.

7.1. Summary and Conclusions

In this thesis we introduced a concept for a multi-device notification framework. First we discussed related work regarding notifications and interruptions caused by notifications. Studies have shown that notifications are disruptive and lower task performance [6, 7, 26] but are still valued by users [24]. We then looked at the recent research regarding mobile notifications and described the *Desktop Notifications* [36] application, which allows sending notifications from Android-based devices to desktop computers. Afterwards we described various notification modalities and related research. Furthermore, we discussed the current implementation of notifications in popular desktop and mobile operating systems, notifications in the web and the state of synchronizing notifications across devices. We closed the section regarding related work with a selection of recently announced devices that are able to notify the user.

Based on the described related work and the current implementation of notifications on different platforms we proposed our concept of a multi-device notification framework. The goal of the framework is the synchronization of notifications across multiple devices. A notification on one device should also be shown on all other connected devices and dismissing a notification on one device should remove it from all others. As a starting point for the implementation of the framework we used the *Desktop Notifications* application. Before the actual implementation, we conducted two studies with the existing user base of the application to determine what kind of devices the users own and on which devices they would like to receive notifications. The results of the first study showed that smartphones, tablets, desktop PCs, laptops and TVs are common, while smartwatches and smartglasses are not yet popular. According to the results of the second survey, the users agreed with the idea of receiving notifications on different kinds of devices with the exception of TVs. Based on these findings we implemented the framework for smartphones, tablets and desktop computers by updating the *Desktop Notifications* application. We then distributed the updated application to the existing user base. After two months over 33,000 users were actively using the updated application.

Using the data collected by the application, we analyzed which categories of apps were disabled from sending notifications through our app most frequently. Our results show that most of these apps were from the *tool* category. The app with the highest exclusion percentage was disabled by 9.9% of all users. In the same manner we extracted the categories of apps that were set to “Private Mode” most frequently, finding that most were apps related to communication. The content of notifications created by these apps often is of private or confidential nature, yet at the same time receiving notifications from these types of apps is important to the users [36]. Not sending the actual content of the notification but still getting notified is a compromise many users seem to agree with.

We then conducted a large-scale study about the usefulness of synchronized notifications. Notifications from the *messenger* category received the highest ratings on all types of devices. The other categories showed mixed results, with the desktop receiving the overall higher ratings compared to smartphones and tablets. We assume that the users received duplicate notifications on the latter two devices thus lowering the usefulness of the synchronized notifications. However, the results could also be influenced by the name of the application (*Desktop Notifications*). On the basis of these mixed results we updated the application to allow disabling the sending of notifications by device type instead of disabling them completely. Again we were able to observe that apps from the *tool* category were excluded from sending notifications most frequently on all types of devices.

7.2. Future Work

The research-in-the-large approach of deploying our application in an app store helped us to gain real-world insights into user preferences and behavior with notifications. However, this approach can be a challenge for less popular device types and ones with a large number of different platforms, for example smartwatches and smartglasses. A follow-up study in a controlled environment would allow us to cover these kinds of devices.

While our framework allows the synchronization of notifications across multiple devices, there is no user interaction possible on receiving devices except the dismissal of said notifications. However, nowadays most notifications are interactive and allow the user to take immediate action. Therefore, future work should explore how other actions relating to notifications can be added to the framework. Considering on the variety of input mechanisms between devices, this can be a challenging task.

Previous work has shown that contextual information can be used to avoid disruptive notifications [25]. By delaying a notification until an appropriate moment occurs, like the end of a task, it is possible to reduce the likelihood or severity of interruption caused by the notification. This has already been explored for single-device environments [10, 19, 21, 23, 30]. Delaying a notification is a balancing act between the importance of said notification and finding an

appropriate moment. It should be investigated how this work can be incorporated into a multi-device environment. Our framework is based on the idea of broadcasting events to all devices. Later, we modified the implementation to allow the disabling of sending notifications to certain device types. It should be explored if it is possible to automatically determine the most suitable device or subset of devices for a given context and notification. This is especially challenging for mobile devices. Prior work has already investigated how mobile phones can infer where they are kept [40]. Furthermore, many devices support multiple notification modalities to alert the user. Again, based on the current context and importance of the notification, used modalities should be used in degrees, ranging from subtle to intrusive. In summary, it should be explored when, on which devices and by what means the user should be notified. Each of these dimensions requires a model that can predict the importance of notifications and a shared context between the user's devices.

A. Appendix

A.1. Package Names and Categories

Package names and corresponding categories of the apps used in the sections 4.2 and 6.4.

Package Name ▼	Category
com.android.calendar	calendar/alarm
com.android.deskclock	calendar/alarm
com.android.email	email
com.android.mms	messenger
com.android.phone	phone
com.android.providers.downloads	tool
com.android.settings	tool
com.android.systemui	tool
com.android.vending	tool
com.antivirus	tool
com.bbm	messenger
com.cleanmaster.mguard	tool
com.cleanmaster.security	tool
com.dianxinos.optimizer.duplay	tool
com.facebook.katana	social
com.facebook.orca	messenger
com.foursquare.robin	social
com.fsck.k9	email
com.google.android.apps.plus	social

(continued on the next page)

A. Appendix

Package Name ▼	Category
com.google.android.calendar	calendar/alarm
com.google.android.deskclock	calendar/alarm
com.google.android.email	email
com.google.android.gm	email
com.google.android.googlequicksearchbox	tool
com.google.android.talk	messenger
com.google.android.youtube	social
com.htc.android.mail	email
com.htc.sense.mms	messenger
com.ijinshan.kbatterydoctor_en	tool
com.instagram.android	social
com.jb.gosms	messenger
com.mailboxapp	email
com.outlook.Z7	email
com.qihoo.security	tool
com.sec.android.app.clockpackage	tool
com.skype.raider	messenger
com.snapchat.android	messenger
com.sonyericsson.conversations	messenger
com.supercell.clashofclans	game
com.twitter.android	social
com.viber.voip	messenger
com.whatsapp	messenger
com.yahoo.mobile.client.android.mail	email
jp.naver.line.android	messenger
kik.android	messenger
org.telegram.messenger	messenger

A.2. Survey about Notifications on Multiple Devices (Full Size)

 Survey

You received a notification from **Spotify**. Assume you own **all** of the following devices.

Please rate the following six statements using this scale.

1 = Strongly agree
2 = Agree
3 = Neutral
4 = Disagree
5 = Strongly disagree

I would like to see this notification on my **desktop PC**.

1 2 3 4 5

Strongly agree Strongly disagree

I would like to see this notification on my **tablet**.

1 2 3 4 5

Strongly agree Strongly disagree

I would like to see this notification on my **smartwatch**.

1 2 3 4 5

Strongly agree Strongly disagree

I would like to see this notification on my **laptop**.

1 2 3 4 5

Strongly agree Strongly disagree

I would like to see this notification on my **smartphone**.

1 2 3 4 5

Strongly agree Strongly disagree

I would like to see this notification on my **TV**.

1 2 3 4 5

Strongly agree Strongly disagree

I would like to see this notification on my **smartglasses**.

1 2 3 4 5

Strongly agree Strongly disagree

Comment (optional)

Submit

The survey only collects your answer and the name of the app. The content of the notification is not sent.

By answering this questions you help our research. If you have any questions contact us at hclilab@gmail.com.

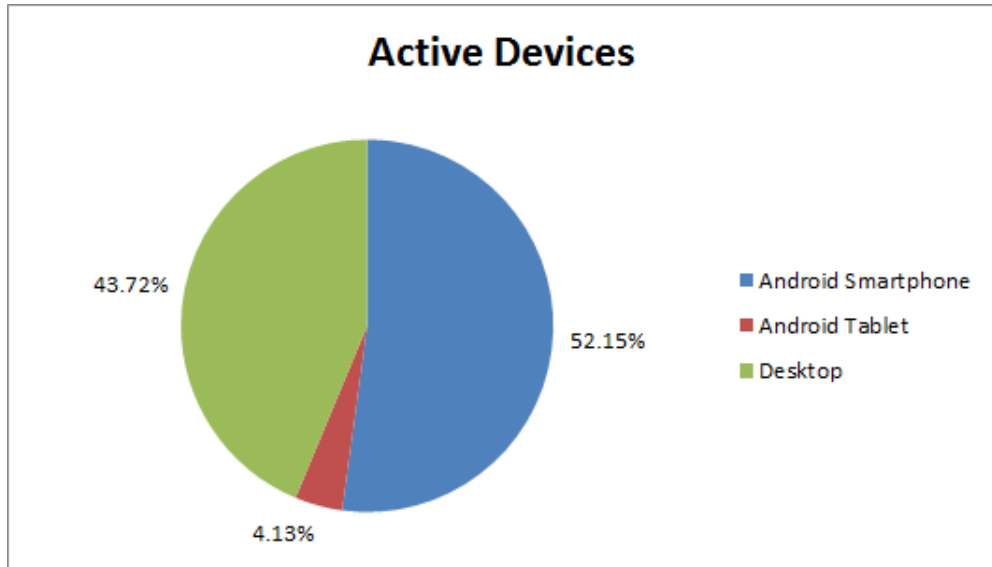
A.3. Most Frequently Disabled Apps

Package Name	Category	Percent Disabled ▼
com.google.android.googlequicksearchbox	tool	9.90%
com.keramidas.TitaniumBackup	tool	8.43%
com.fsck.k9	email	8.40%
com.android.vending	tool	8.02%
com.antivirus	tool	7.70%
de.robv.android.xposed.installer	tool	7.53%
com.touchtype.swiftkey	tool	6.84%
com.google.android.email	email	6.83%
com.ninegag.android.app	media	6.68%
com.cleanmaster.mguard	tool	6.62%
com.nero.android.htc.sync	tool	6.48%
com.sonyericsson.extras.liveware	tool	6.03%
com.htc.android.mail	email	6.02%
com.ea.games.r3_row	game	5.86%
com.sec.android.fwupgrade	tool	5.73%
com.vkontakte.android	social	5.68%
com.estrongs.android.pop	tool	5.67%
com.sony.nfx.app.sfr	tool	5.64%
com.amazon.venezia	tool	5.58%
com.google.android.apps.maps	tool	5.55%
com.kiloo.subwaysurf	tool	5.46%
mobi.mgeek.TunnyBrowser	tool	5.34%
com.android.providers.downloads	tool	5.27%
com.sonymobile.playanywhere	tool	5.21%
com.motorola.contextual.smartrules2	tool	5.18%
com.google.android.gm	email	5.15%
com.amazon.kindle	media	5.12%
com.cleanmaster.security	tool	5.12%
com.king.candycrushsaga	game	5.12%
com.ifttt.ifttt	tool	5.07%

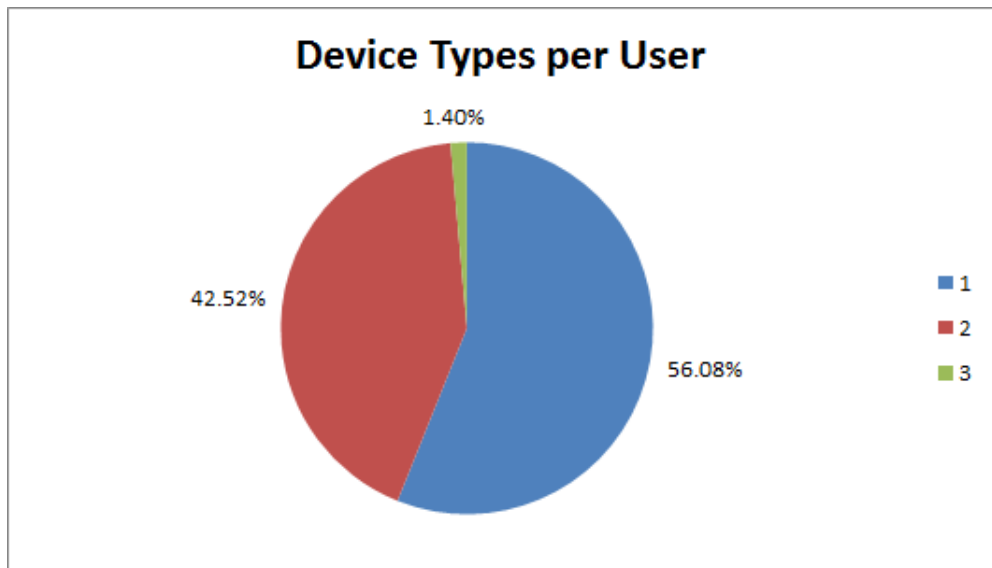
A.4. Most Private Apps

Package Name	Category	Percent Private ▼
com.fsck.k9	email	1.72%
ch.threema.app	messenger	1.39%
com.bbm	messenger	1.13%
com.whatsapp	messenger	1.04%
kik.android	messenger	0.99%
com.htc.sense.mms	messenger	0.85%
org.telegram.messenger	messenger	0.72%
com.tencent.mm	messenger	0.61%
com.google.android.talk	messenger	0.56%
com.android.mms	messenger	0.53%
com.sonyericsson.conversations	messenger	0.49%
com.jb.gosms	messenger	0.47%
com.google.android.gm	email	0.45%
com.yahoo.mobile.client.android.mail	email	0.43%
com.htc.android.mail	email	0.43%
com.facebook.orca	messenger	0.43%
com.google.android.email	email	0.38%
cm.aptoide.pt	tool	0.34%
com.skype.raider	messenger	0.34%
com.facebook.katana	social	0.33%
com.outlook.Z7	email	0.33%
com.viber.voip	messenger	0.32%
com.android.email	email	0.32%
com.paypal.android.p2pmobile	tool	0.28%
jp.naver.line.android	messenger	0.27%
com.antivirus	tool	0.26%
com.contapps.android	social	0.25%
com.cleanmaster.security	tool	0.25%
com.htc.calendar	calendar/alarm	0.23%
com.snapchat.android	messenger	0.23%

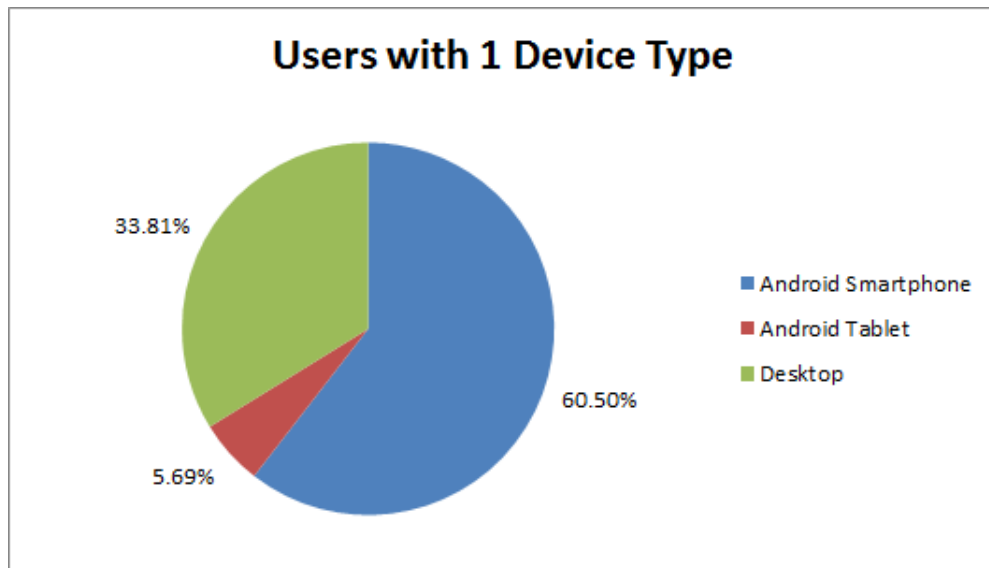
A.5. Device and Language Distribution



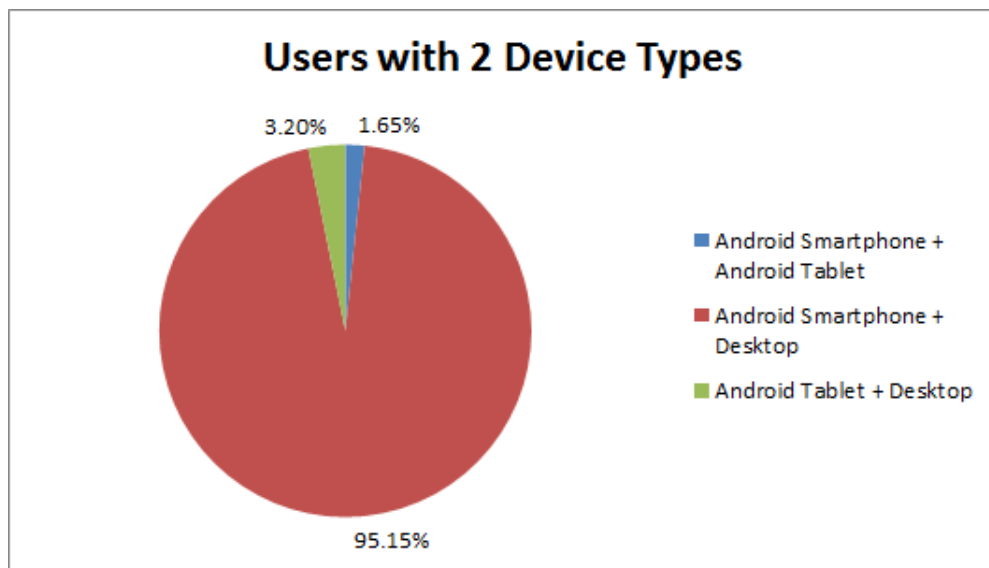
Active devices by device type (N = 53,565)



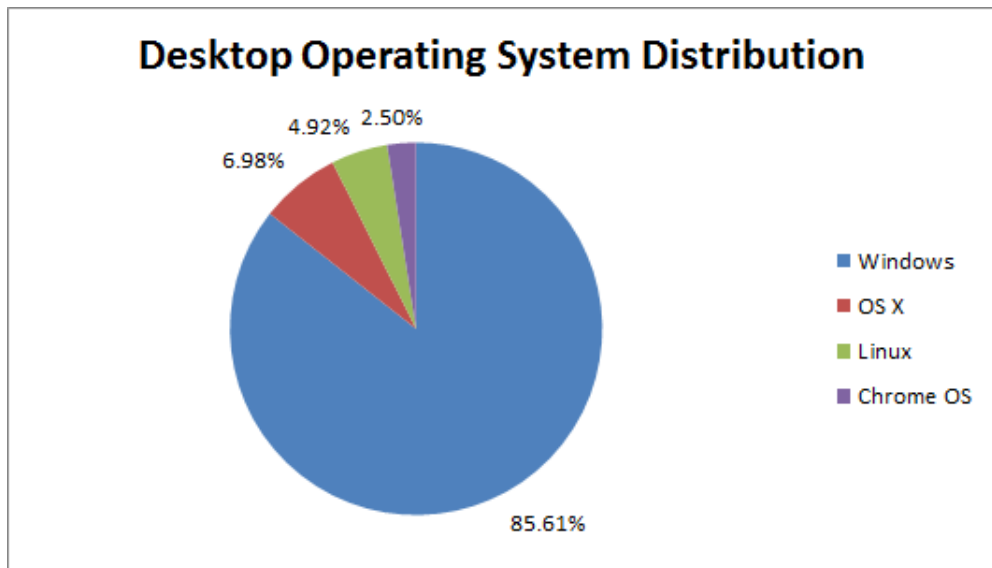
Device types per user (N = 33,888)



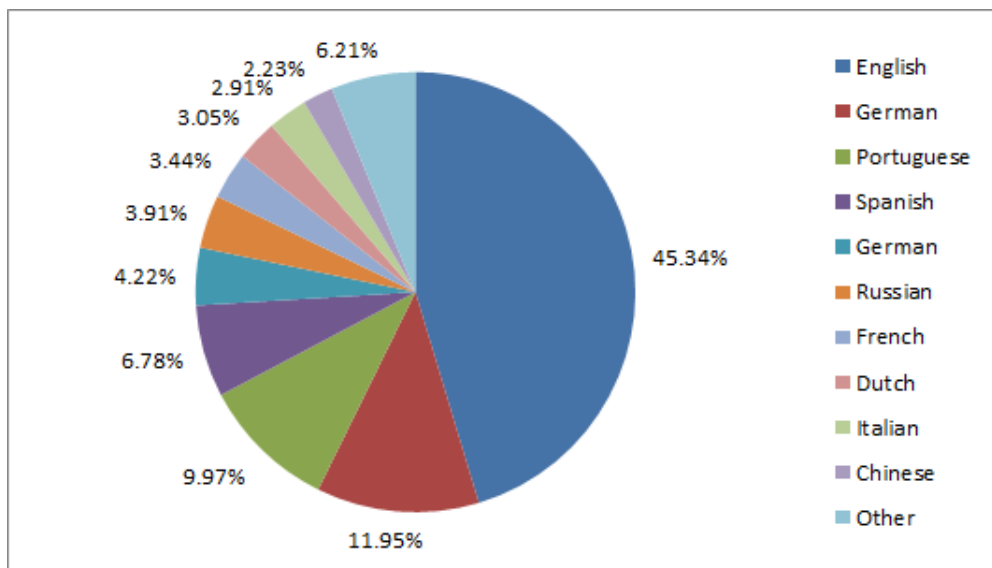
Distribution of users with one device type (N = 19,004)



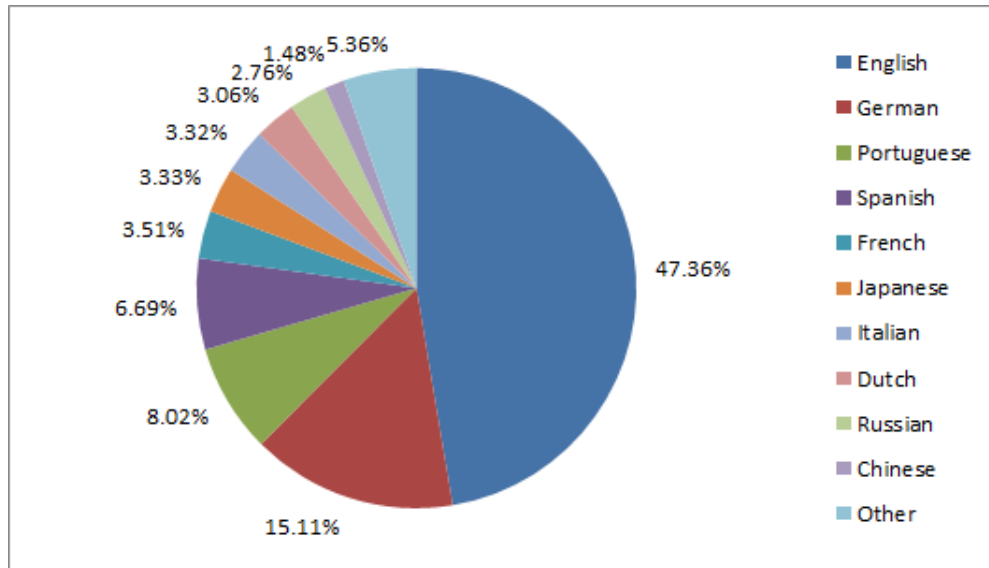
Distribution of users with two device types (N = 14,409)



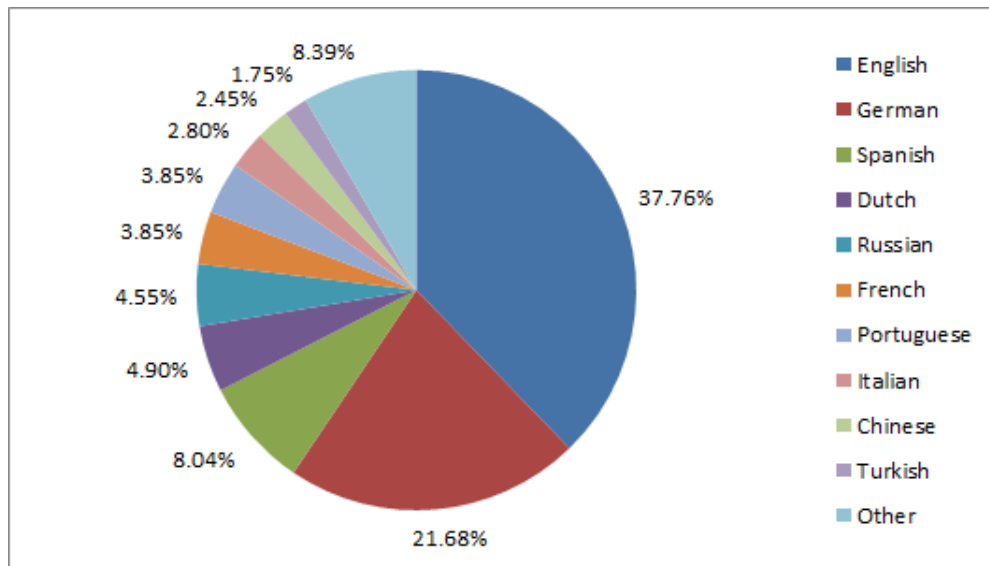
Desktop operating system distribution (N = 23,418)



Language distribution in section 4.1 (N = 6,779)



Language distribution in section 4.2 (N = 6,961)



Language distribution in section 6.4 (N = 286)

A.6. Updated Most Frequently Disabled Apps (Smartphone)

Package Name	Category	Percent Disabled ▼
com.keramidas.TitaniumBackup	category	2.92%
com.google.android.googlequicksearchbox	tool	2.69%
de.robv.android.xposed.installer	tool	2.69%
com.google.android.apps.inbox	tool	2.28%
com.motorola.contextual.smartrules2	email	2.22%
com.google.android.apps.books	tool	2.20%
com.android.vending	media	2.20%
com.nianticproject.ingress	tool	2.15%
eu.chainfire.supersu	tool	2.13%
com.amazon.venezia	tool	2.13%
com.google.android.apps.paidtasks	tool	2.02%
com.google.android.music	tool	1.91%
com.google.android.gm	media	1.91%
com.estrongs.android.pop	email	1.90%
com.google.android.calendar	tool	1.79%
com.touchtype.swiftkey	calendar/alarm	1.77%
com.google.android.email	tool	1.77%
com.sony.nfx.app.sfr	email	1.67%
com.amazon.mShop.android	media	1.65%
com.sonymobile.playanywhere	tool	1.64%
com.android.deskclock	tool	1.58%
com.cleanmaster.security	calendar/alarm	1.58%
com.android.providers.downloads	tool	1.57%
com.pushbullet.android	tool	1.53%
com.dropbox.android	tool	1.48%
com.amazon.kindle	tool	1.45%
com.ninegag.android.app	media	1.44%
com.google.android.apps.fitness	media	1.44%
com.google.android.deskclock	tool	1.43%
tunein.player	calendar/alarm	1.42%

A.7. Updated Most Frequently Disabled Apps (Tablet)

Package Name	Category	Percent Disabled ▼
com.keramidas.TitaniumBackup	tool	3.09%
com.google.android.googlequicksearchbox	tool	2.82%
de.robv.android.xposed.installer	tool	2.69%
com.motorola.contextual.smartrules2	tool	2.50%
com.amazon.venezia	tool	2.39%
com.android.vending	tool	2.33%
com.google.android.apps.inbox	email	2.33%
com.google.android.apps.paidtasks	tool	2.28%
com.google.android.apps.books	media	2.25%
eu.chainfire.supersu	tool	2.18%
com.nianticproject.ingress	tool	2.05%
com.google.android.gm	email	2.02%
com.estrongs.android.pop	tool	1.96%
com.google.android.music	media	1.95%
com.google.android.email	email	1.95%
com.touchtype.swiftkey	tool	1.90%
com.google.android.calendar	calendar/alarm	1.87%
com.amazon.mShop.android	tool	1.80%
com.sony.nfx.app.sfrs	media	1.67%
com.android.providers.downloads	tool	1.64%
com.sonymobile.playanywhere	tool	1.64%
com.pushbullet.android	tool	1.64%
com.amazon.kindle	media	1.61%
com.dropbox.android	tool	1.58%
com.android.deskclock	calendar/alarm	1.58%
com.cleanmaster.security	tool	1.58%
com.ijinshan.kbatterydoctor_en	tool	1.54%
tunein.player	tool	1.52%
com.musixmatch.android.lyrify	media	1.50%
com.google.android.deskclock	calendar/alarm	1.48%

A.8. Updated Most Frequently Disabled Apps (Desktop)

Package Name	Category	Percent Disabled ▼
com.keramidas.TitaniumBackup	tool	2.92%
com.google.android.googlequicksearchbox	tool	2.90%
de.robv.android.xposed.installer	tool	2.36%
eu.chainfire.supersu	tool	2.33%
com.amazon.venezia	tool	2.33%
com.android.vending	tool	2.22%
com.motorola.contextual.smartrules2	tool	2.02%
com.google.android.apps.inbox	email	1.98%
com.google.android.apps.books	media	1.87%
com.touchtype.swiftkey	tool	1.87%
com.musixmatch.android.lyrify	media	1.80%
com.google.android.email	email	1.77%
com.google.android.music	media	1.76%
com.estrongs.android.pop	tool	1.72%
com.sony.nfx.app.sfr	media	1.67%
com.google.android.gm	email	1.66%
com.android.providers.downloads	tool	1.65%
com.amazon.mShop.android	tool	1.65%
com.cleanmaster.security	tool	1.63%
com.lookout	tool	1.55%
com.amazon.kindle	media	1.53%
com.sonyericsson.extras.liveware	tool	1.52%
com.htc.android.mail	email	1.48%
com.king.candycrushsaga	game	1.48%
com.antivirus	tool	1.47%
com.dropbox.android	tool	1.46%
com.sonymobile.playanywhere	tool	1.46%
com.cleanmaster.mguard	tool	1.45%
com.ninegag.android.app	media	1.44%
tunein.player	tool	1.42%

Bibliography

- [1] P. D. Adamczyk and B. P. Bailey. 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*, pages 271–278. ACM, 2004. (Cited on page 15)
- [2] R. Berjon, S. Faulkner, T. Leithead, S. Pfeiffer, E. O’Connor, and E. D. Navara. HTML5. Candidate recommendation, W3C, July 2014. URL <http://www.w3.org/TR/2014/CR-html5-20140731/>. (Cited on page 23)
- [3] C. S. Campbell and P. Tarasewich. Designing visual notification cues for mobile devices. In *CHI’04 Extended Abstracts on Human Factors in Computing Systems*, pages 1199–1202. ACM, 2004. (Cited on page 19)
- [4] K. Church and R. de Oliveira. What’s up with whatsapp?: comparing mobile instant messaging behaviors with traditional sms. In *Proceedings of the 15th international conference on Human-computer interaction with mobile devices and services*, pages 352–361. ACM, 2013. (Cited on page 16)
- [5] E. Cutrell, M. Czerwinski, and E. Horvitz. Notification, disruption, and memory: Effects of messaging interruptions on memory and performance. 2001. (Cited on page 15)
- [6] M. Czerwinski, E. Cutrell, and E. Horvitz. Instant messaging and interruption: Influence of task type on performance. In *OZCHI 2000 conference proceedings*, volume 356, page 361, 2000. (Cited on pages 13, 15, 26 and 65)
- [7] M. Czerwinski, E. Horvitz, and S. Wilhite. A diary study of task switching and interruptions. In *Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 175–182. ACM, 2004. (Cited on pages 13, 15, 26 and 65)
- [8] Ecma International. The json data interchange format. URL <http://www.ecma-international.org/publications/files/ECMA-ST/ECMA-404.pdf>, 2013. (Cited on page 39)
- [9] A. P. Felt, S. Egelman, and D. Wagner. I’ve got 99 problems, but vibration ain’t one: a survey of smartphone users’ concerns. In *Proceedings of the second ACM workshop on Security and privacy in smartphones and mobile devices*, pages 33–44. ACM, 2012. (Cited on page 28)

- [10] J. E. Fischer, C. Greenhalgh, and S. Benford. 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*, pages 181–190. ACM, 2011. (Cited on page 66)
- [11] J. E. Fischer, N. Yee, V. Bellotti, N. Good, S. Benford, and C. Greenhalgh. Effects of content and time of delivery on receptivity to mobile interruptions. In *Proceedings of the 12th international conference on Human computer interaction with mobile devices and services*, pages 103–112. ACM, 2010. (Cited on page 15)
- [12] A. Gomes, A. Nesbitt, and R. Vertegaal. Morephone: a study of actuated shape deformations for flexible thin-film smartphone notifications. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pages 583–592. ACM, 2013. (Cited on page 19)
- [13] R. Hansson and P. Ljungstrand. The reminder bracelet: subtle notification cues for mobile devices. In *CHI'00 extended abstracts on Human factors in computing systems*, pages 323–324. ACM, 2000. (Cited on page 18)
- [14] R. Hansson, P. Ljungstrand, and J. Redström. Subtle and public notification cues for mobile devices. In *UbiComp 2001: Ubiquitous Computing*, pages 240–246. Springer, 2001. (Cited on page 18)
- [15] F. Hemmert, S. Hamann, M. Löwe, J. Zeipelt, and G. Joost. Shape-changing mobiles: tapering in two-dimensional deformational displays in mobile phones. In *CHI'10 Extended Abstracts on Human Factors in Computing Systems*, pages 3075–3080. ACM, 2010. (Cited on page 19)
- [16] N. Henze and M. Pielot. App stores: external validity for mobile hci. *interactions*, 20(2):33–38, 2013. (Cited on page 13)
- [17] N. Henze, M. Pielot, B. Poppinga, T. Schinke, and S. Boll. My app is an experiment: Experience from user studies in mobile app stores. *International Journal of Mobile Human Computer Interaction (IJMHCI)*, 3(4):71–91, 2011. (Cited on page 13)
- [18] I. Hickson. The websocket api. Candidate recommendation, W3C, Sept. 2012. URL <http://www.w3.org/TR/2012/CR-websockets-20120920/>. (Cited on page 53)
- [19] J. Ho and S. S. Intille. 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*, pages 909–918. ACM, 2005. (Cited on page 66)
- [20] M. Honan. Why notifications are about to rule the smartphone interface. URL <http://www.wired.com/2014/06/smartphone-notifications/>, June 2014. (Cited on page 18)

-
- [21] E. Horvitz, A. Jacobs, and D. Hovel. Attention-sensitive alerting. In *Proceedings of the Fifteenth conference on Uncertainty in artificial intelligence*, pages 305–313. Morgan Kaufmann Publishers Inc., 1999. (Cited on page 66)
- [22] Internet Engineering Task Force (IETF). The oauth 2.0 authorization framework. URL <http://tools.ietf.org/html/rfc6749>, 2012. (Cited on page 40)
- [23] S. T. Iqbal and B. P. Bailey. Oasis: A framework for linking notification delivery to the perceptual structure of goal-directed tasks. *ACM Transactions on Computer-Human Interaction (TOCHI)*, 17(4):15, 2010. (Cited on page 66)
- [24] S. T. Iqbal and E. Horvitz. Notifications and awareness: a field study of alert usage and preferences. In *Proceedings of the 2010 ACM conference on Computer supported cooperative work*, pages 27–30. ACM, 2010. (Cited on pages 16, 26 and 65)
- [25] J. Knittel, A. Sahami Shirazi, N. Henze, and A. Schmidt. Utilizing contextual information for mobile communication. In *CHI'13 Extended Abstracts on Human Factors in Computing Systems*, pages 1371–1376. ACM, 2013. (Cited on page 66)
- [26] L. Leiva, M. Böhmer, S. Gehring, and A. Krüger. Back to the app: the costs of mobile application interruptions. In *Proceedings of the 14th international conference on Human-computer interaction with mobile devices and services*, pages 291–294. ACM, 2012. (Cited on pages 13, 15, 26 and 65)
- [27] G. Mark, S. Voida, and A. Cardello. A pace not dictated by electrons: an empirical study of work without email. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pages 555–564. ACM, 2012. (Cited on page 15)
- [28] A. Mashhadi, A. Mathur, and F. Kawsar. The myth of subtle notifications. In *Proceedings of the 2014 ACM International Joint Conference on Pervasive and Ubiquitous Computing: Adjunct Publication*, pages 111–114. ACM, 2014. (Cited on page 18)
- [29] D. McMillan, A. Morrison, O. Brown, M. Hall, and M. Chalmers. Further into the wild: Running worldwide trials of mobile systems. In *Pervasive Computing*, pages 210–227. Springer, 2010. (Cited on page 13)
- [30] V. Pejovic and M. Musolesi. Interruptme: designing intelligent prompting mechanisms for pervasive applications. In *Proceedings of the 2014 ACM International Joint Conference on Pervasive and Ubiquitous Computing*, pages 897–908. ACM, 2014. (Cited on page 66)
- [31] M. Pielot. Large-scale evaluation of call-availability prediction. In *Proceedings of the 2014 ACM International Joint Conference on Pervasive and Ubiquitous Computing*, pages 933–937. ACM, 2014. (Cited on page 16)
- [32] M. Pielot, K. Church, and R. de Oliveira. An in-situ study of mobile phone notifications. In *Proceedings of the 16th international conference on Human-computer interaction with mobile devices & services*, pages 233–242. ACM, 2014. (Cited on page 16)

- [33] M. Pielot, R. de Oliveira, H. Kwak, and N. Oliver. Didn't you see my message?: predicting attentiveness to mobile instant messages. In *Proceedings of the 32nd annual ACM conference on Human factors in computing systems*, pages 3319–3328. ACM, 2014. (Cited on page 16)
- [34] A. Russell and J. Song. Service workers. W3C working draft, W3C, May 2014. URL <http://www.w3.org/TR/2014/WD-service-workers-20140508/>. (Cited on page 23)
- [35] A. Sahami, P. Holleis, A. Schmidt, and J. Häkkinä. Rich tactile output on mobile devices. In *Ambient Intelligence*, pages 210–221. Springer, 2008. (Cited on page 19)
- [36] A. Sahami Shirazi, N. Henze, T. Dingler, M. Pielot, D. Weber, and A. Schmidt. Large-scale assessment of mobile notifications. In *Proceedings of the 32nd annual ACM conference on Human factors in computing systems*, pages 3055–3064. ACM, 2014. (Cited on pages 13, 17, 24, 26, 27, 28, 31, 39, 55, 58, 61, 65 and 66)
- [37] P. Tarasewich, T. Bhimdi, and M. Dideles. Testing visual notification cues on a mobile device. In *CHI'04 Extended Abstracts on Human Factors in Computing Systems*, pages 1562–1562. ACM, 2004. (Cited on page 19)
- [38] A. van Kesteren and J. Gregg. Web notifications. Last call WD, W3C, Sept. 2013. URL <http://www.w3.org/TR/2013/WD-notifications-20130912/>. (Cited on page 23)
- [39] D. Warnock, M. McGee-Lennon, and S. Brewster. Multiple notification modalities and older users. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pages 1091–1094. ACM, 2013. (Cited on page 19)
- [40] J. Wiese, T. S. Saponas, and A. Brush. Phoneprioception: enabling mobile phones to infer where they are kept. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pages 2157–2166. ACM, 2013. (Cited on page 67)
- [41] S. Yamanaka and H. Miyashita. Vibkinesis: notification by direct tap and'dying message'using vibronic movement controllable smartphones. In *Proceedings of the 27th annual ACM symposium on User interface software and technology*, pages 535–540. ACM, 2014. (Cited on page 18)

All links were last followed on January 14, 2015.

Declaration

I hereby declare that the work presented in this thesis is entirely my own and that I did not use any other sources and references than the listed ones. I have marked all direct or indirect statements from other sources contained therein as quotations. Neither this work nor significant parts of it were part of another examination procedure. I have not published this work in whole or in part before. The electronic copy is consistent with all submitted copies.

place, date, signature