Automatic check-out reminder system for public transport in Copenhagen

Simon Gray
IT University of Copenhagen
Rued Langgaards Vej 7, Kbh S
simg@itu.dk

Troels Siggaard
IT University of Copenhagen
Rued Langgaards Vej 7, Kbh S
tsig@itu.dk

ABSTRACT
Research and public opinion have shown that users of the Danish travel card system, Rejsekort, often pay fines for forgetting to check out with the card. This paper presents an automatic, context-aware system to help remind people to check out. We have implemented the system as a service running on the Android operating system, using native Android notifications reminding people to check out, triggered according to a rule-based logic. We evaluated the accuracy of the application as well as documented current technical issues and user opinion. This evaluation showed that the automatic check-out concept was viable for typical user behavior, but could result in unpredictable results in atypical situations.

Author Keywords
Context awareness, public transport, android, reminder

INTRODUCTION
Public transport plays an important role in any modern metropolitan infrastructure. This importance has been broadly recognized, for example at the G20 summit in Mexico in 2012 [16].

The quest for an ever more reliable and efficient public transport system has brought with it new technologies to refine existing transportation systems. Around the beginning of the new millennium, several public transport companies around the world started introducing electronic ticketing systems. The Oyster card in London, England and Rejsekort (Travel Card) in Denmark were both inspired by the RFID-based electronic ticketing system, the Octopus card, which was implemented in Hong Kong in 1997 [15].

The public transport companies DSB, Ørestadsselskabet, and Movia started collaborating on the idea of creating an electronic travel card in the year 1999. Later, this collaboration spawned the company Rejsekort A/S for the task of developing an electronic travel card for use in all of Denmark. [14]

The purpose of the Rejsekort travel card is to make a flexible ticketing system and to give the Danes an easy and simple way of travelling throughout the country using trains and busses. [12] Rejsekort had difficulties in implementing the system and the countrywide launch was delayed and instead made incremental[9]. By late 2012, the system was fully operational in Denmark and the cost of the system ended up being 1.3 billion Danish kroner. [13]

Design issues of Rejsekort
With Rejsekort, the public transit user should no longer buy tickets or stamp ten-trip cards, but instead remember to check in at a terminal each time he/she boards a new type of public transport and check out once the journey ends. The item used to identify the user's account is a small plastic card with a RFID chip.[1]

Unfortunately, the Rejsekort system has been plagued with several major design issues [2, 3, 4, 5].

The check-out issue
One of the most common sources of irritation have to with the two or more steps of checking in and checking out with Rejsekort. Instead of a single check-in in the form of a ticket purchase or stamping the ten-trip card, the user now has to remember to also check in every time he/she gets on a new type of public transport and, crucially, to check out; otherwise, the user will be charged a fee of 50 Danish kroner. This has been a recurring issue for users of the system since its introduction [6]. Apart from the fees, the system also burdens the user with added stress as it is completely up to the user to remember to check out, the system does not remind him/her.

Structure of this paper
The rest of the paper comprises the following topics

- An overview of existing approaches to mitigating the check-out issue
- An introduction to our alternative check-out reminder
- Related academic work
- Presentation of our iterative development process
- Overview of the system
- Evaluation of the system
- Discussion of evaluation results
- Conclusion

Existing solutions to solve the check-out issue
In attempt to mitigate the check-out issue, Danish developers have created mobile phone apps to remind Rejsekort users to check out in time [7].

These apps work by requiring an additional step of typing in the name of the destination, then creating a geofence[1] around it that triggers a mobile phone notification reminding the user to check out. This helps to solve the issue with the fees, but does not reduce stress as they introduce yet another step to the process of travelling with public transport.

Name	Platform	Requirements	User involvement	Steps	Input destination	Works in all of Denmark
Tjek Ud Appen	Android	GPS Internet	Yes, type destination	3 - 4	Yes, type destination	Yes
Check Ud -Rejsekort	iOS	GPS Internet	Yes, type destination	2 - 3	Yes, type destination	Yes
Check Ud	iOS	GPS Internet	Yes, type destination	3 - 4	Yes, type destination	Yes
TjekMate	iOS	GPS Internet	Yes, type destination	3 - 4	Yes, type destination	Yes
Our solution	Android	Accelerometer Wi-Fi GSM (cellular)	No *	0 - 2 **	No	No, only Copenhagen

Table 1: Our approach versus existing apps

Our alternative approach – system requirements

Instead of introducing an additional step when travelling, a more elegant way to implement check-out reminders would be to create a system that:

- Is fully automatic
- Is context-aware
- Has no need for any dedicated infrastructure
- Does not use GPS
- Has an easy fallback method for manual input

Such a system would:

- Not introduce any more unnecessary stress
- Solve the issue with "forgotten check-out"-fees
- Not drain the battery excessively

Exploring the automatic reminder concept

Instead of a traditional focus group, we conducted a virtual meeting on http://reddit.com/r/denmark - a website where user comments are ranked based on how many up-votes and down-votes they receive from other users.

We asked what users thought of an always-on service on their phones which would, among other features related to public transport, automatically detect travels with public transport and could deliver notifications right after getting off [17].

- Several users thought it was a good idea and the topic itself attracted a combined score of 38 points, which is the mark of a high interest topic.
- The most up voted comment had concerns about the potential battery usage of the app.
- One user thought the data produced by the app would be of interest to public transport companies, but had reservations when it came to privacy.
- A few users did not understand the purpose of the app.
- A more technically minded user foresaw issues with false positives and missing Wi-Fi signals on busses, but thought the idea was good.

RELATED WORK

In this related work section we first take a look at a usability study of Rejsekort and the existing solutions to solve the check-out issue. Then we draw on prior academic research, to inform us on the approaches to tracking people in public transit. We will introduce several technologies and their tradeoffs, when used in context-aware or mobile systems.

Heuristic evaluation of the check-out issue

An issue of Interaction magazine had an in-depth article documenting the usability of the Rejsekort system [18]. In this article, the reactions of six first-time users of Rejsekort were studied when using the card.

The study touches on the inconsistency between the existing ten-trip card and ticket systems, which unlike Rejsekort only require a check in, but no check out. There are stickers placed close to the doors and check-out terminals, but they do not attract much attention. One of the six users in the study had trouble remembering to check out.

The study makes the following comments about the check-out issue:

"If a person forgets to check out and leaves the station or bus stop, he/she will not receive any notification"

Differentiating modes of transportation

Boriel et al. [10] tracks transport modes using location and sensory data. They analyzed a week's worth of sensory data and compared to ground truth to suggest sensory patterns that differentiate different transport modes.

They suggest ways to differentiate between walking and cycling. Walking or cycling is identified via periodic movements detected using the accelerometer and GPS speed measurements is used to differentiate between the two. Differentiating between car and bus is difficult, though, as both the accelerometer data and speed data is largely the same. They suggest overlaying a GPS trace on a street map grid and comparing to known bus routes. To

[1] Geofence is a geographical radius of a location.

further differentiate from the sensor pattern of cars, they also suggest comparing traffic stops to known locations of bus stops.

We see, in this particular article, a trend of relying mainly on data mining analysis based on one type of technology like accelerometers etc.. Data mining often takes a lot of effort due to the amount of data needed for activity recognition, it sometimes requires one to make use of advanced methods and algorithms to have a high enough recognition rate to be usable. [10]

Smartphones, sensors and Android

Google has released an API alongside their v4.3 of the Android operating system, which makes it easy to get Activity Recognition results from the accelerometer in the smartphone [8]. Besides accelerometer, the technologies usually found in Android smartphones are Wi-Fi, Bluetooth, GSM/UMTS/LTE, light sensor, camera, microphone, proximity sensor and NFC. App developers can thereby collect different kinds of data and use these data i.e. context-aware apps.

Generally the sensors are accurate enough to gather useful data, but [21] found that Wi-Fi signal strength above -80 dB is generally needed to obtain a reliable result on Android to reliably determine location indoor [21].

Context-aware systems and location as a context

Context-aware systems usually rely on absolute location often acquired with use of GPS [21]. One of the tradeoffs while using GPS is battery life, as noted in [22]. GPS is the most battery-intensive technology compared to Accelerometer, GSM, Bluetooth and Wi-Fi. Another trade-off is the line of sight characteristics of GPS, which limits use inside, in tunnels and sometimes even getting a satellite fix inside vehicles.

Another approach to gain information of location is by using other technologies installed around i.e. a city, this could be Wi-Fi or cellular towers. If one knows the name (SSID) and or MAC address (BSSID) of a particular place it's easy to get this location just by continuous scanning. This relative location is possible to use, combined with other sensor result to derive the possible context of a user. Cell Towers etc. might be used to reason about the context i.e. mode of transportation or other locations [about-context-awareness].

Wi-Fi fluctuations

A tradeoff from using Wi-Fi is lack of precision due to the characteristics of radio signals. Wi-Fi signals are affected by a whole range of different things; people-blocking, walls, doors (open/closed), humidity and more [23]. Another tradeoff is that Wi-Fi is very fluctuating while an access point is moving, i.e. Wi-Fi installed in busses. The signal strength can even vary up to 60 dB[11], which is a lot compared to the 8 dB cut in signal that the people-blocking counted for in [23]. Usually Wi-Fi has a range of up to 100-250 meters depending on the wireless technology used and a single access point to draw location from isn't very accurate, compared to the location accuracy from the GPS.

DESIGN PROCESS & METHODS

We divided our development into five iterations, the first one investigating the problem area, the second designing the architecture, the third coding/implementing, testing and evaluating sensors, fourth coding/implementing, testing and evaluating modes and the rule-based logic between them, fifth extensive test, evaluation and bug fixing of the first almost fully working prototype, making quantitative statistical battery tests and releasing it to alpha testers.

Development process

We did incremental prototyping in the first iterations and then proceed with the software development as an evolutionary prototype, building and testing in parallel until we arrived at a final alpha version.

Iterations:

1st iteration:

We started investigating the problem area, found related work, exploring different technologies and devices.

We found the smartphone to be the most popular portable computing device used in Denmark, which made us choose to focus on an app for a smartphone. To adhere to our constraint about using existing infrastructure, the choice of technologies to choose from were limited. Technologies we found useful were Wi-Fi and GSM Cellular Towers in the public transport system and accelerometer.

We found that accelerometer would be a great and battery efficient way to easily get access to activity recognition. Android was an obvious choice, because of the activity recognition API from Google. [8]

Other technologies we found to be interesting were: Bluetooth, Bluetooth LE, NFC/RFID. Unfortunately we didn't find these technologies deployed in the public infrastructure or were restricted access.

GPS was left out as a technology to use, because of its high battery consumption and line of sight characteristic.

We contacted the bus company Movia, and found that not all busses in Copenhagen carry a Wi-Fi access point.

2nd iteration:

We designed the initial architecture and an initial UI mockup of our application, the notifications and widget. The architecture was evaluated and changed to a final one[2].

[2] https://www.dropbox.com/s/cyi0lmsydv68qi4/2013-09-16%2015.49.35.jpg

We started making a quick sketch with pen and paper, of the overall design and user-steps while using our app. The idea and the paper mockup was then refined and graphically designed[3].

We created several use-cases to use while developing the app and held a meeting with Rejsekort A/S to talk about our ideas.

3rd iteration:
We started the software development, coding it while testing in parallel.

We coded the skeleton of our final architecture, and went on to code the service and main activity. Hereafter we started coding the sensors having the use-cases in mind.

We tested the sensor in the public transport system in Copenhagen, activating all sensors, noted their output and fixed several bugs. The output also gave us a good idea of how to design our further app logic.

4th iteration:
This iteration was started by going through the findings from third iteration and build the rule-based logic of our modes of transport. The widget and notifications were implemented.[ref-final-widget] The modes were tested, logic was changed to more accurately determine the correct mode of transport and the transitions between these.

5th iteration
The last iteration we found that we needed a fallback method, an idea we had from the 2nd iteration. This *ForcedMode*, was coded, to let the user change mode of transport if the app wasn't able to determine it automatically. The *ForcedMode* was tested and bugs were found and fixed.

We collected data on how long busses and trains would hold still on a station or bus stops, for use in our apps' rule-based logic.

All of the app including the modes/sensors, widget were extensively tested in a real world setting, travelling with public transport and noting if the logic made was working as intended, by comparing it to the ground truth.
We tested the average time it takes for app to correctly recognize the mode of transport and the average time from leaving the mode of transport until receiving a notification.

Last, we released the app to a handful of potential end users, to test our alpha version of the application and evaluated their experiences with the app.

[3] https://www.dropbox.com/sh/ppkbtsza7996hsj/0BkBRaFwUg

SYSTEM DESIGN

Environmental context-based approach

Rather than making a system that focuses on GPS geofences or data mining and pattern recognition on large sets of sensor data, our geographically limited system exploits existing environmental context within the city of Copenhagen and uses high-level APIs on the Android platform.

Wi-Fi access points transmitting a specific SSID are available in all S-train cars as a means of providing free Internet access to passengers. Similarly, many busses in Copenhagen also come with free Wi-Fi services that have specific SSIDs. The proximity of these SSIDs can be used to sense the presence of S-trains and busses.

In the tunnel section of the Copenhagen metro, cells have been installed to provide passengers with ability to use their mobile phones underground. These cells transmit IDs within a certain range, exhibit a repetitive numeric pattern in different sections of the tunnel, and are not available above ground, which means they can be exploited to sense if a user is inside the tunnel - and also potentially which direction he/she is travelling.

Sensing whether a person is walking or standing still can be used to make context decisions. A continuous walking pattern is not probable inside a bus, for instance. Activity recognition previously had to be done with special hardware and a time-consuming machine learning process, but today we have access to high-level APIs from Google on the Android platform to do this. Additionally, the required hardware sensors, such as accelerometers, are now commonplace in modern smartphones.

Timetables and other statistics for Copenhagen trains and busses can be used to approximate how quickly the system needs to react.

System limitations

The system does not automatically detect travels that:

- Take place solely on the aboveground part of the metro. This is because the metro detection depends isolated cells installed in tunnel section. We did speculate on ways to implement this functionality, but decided not to pursue due to time constraints.
- Take place in busses without Wi-Fi access points. Implementing this functionality seems unlikely, as a possible solution would be a more elaborate implementation of the concept described in the section on "Differentiating modes of transportation" in the Related Academic Work section.
- Take place outside Copenhagen, as local environmental context from other areas will have

to be collected and added to existing or new transportation modes in order to be detected automatically.

However, the user is able to use system in these situations the by explicitly selecting a mode of transportation instead of letting the system detect it automatically. This is described further down in the "User Experience" section.

Rule-based logic

The logic of the system is to constantly cycle between modes of transportation. Once a specific mode of transportation has been reached, fixed rules are in place for when to switch to another mode of transportation. As an example, once the user is detected to be on an S-Train, this is the rule that decides when to leave this mode:

```
protected void evaluate() {
    if (allSignalsLost && !inTunnel) {
        changeMode(ModeTypes.DEFAULT, "");
    }
}
```

Where *allSignalsLost* is a Boolean referring to whether or not we can still detect any S-train wi-fi SSIDS and *inTunnel* is a Boolean referring to whether or not we can detect the cells in the tunnel from Østerport to Vesterport.

This way of switching between modes of transportation results in very clear and succinct code, where the logic can be easily modified after each round of field-testing. The idea has been to keep the rules as simple as possible, until field-testing reveals a more accurate and perhaps more complex set of rules for switching from one mode to another.

Architecture

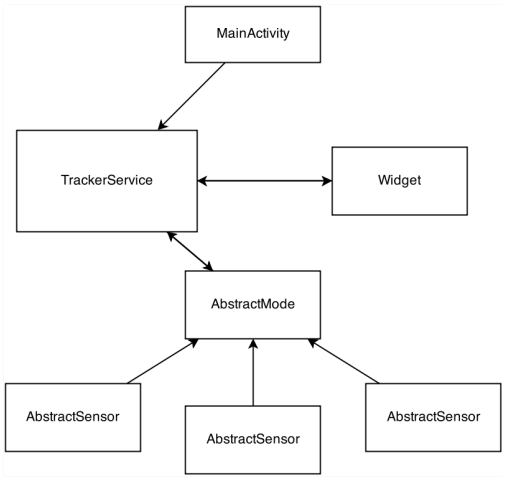

Figure 1: Application architecture

The system[4] has been implemented as an Android application and this section expects the reader to be reasonably familiar with general Android concepts and the Android API. The system architecture uses the observer pattern extensively. The arrows in the above diagram symbolize communication between components.

The **MainActivity** allows the user to launch or shut down the **TrackerService**. This service then keeps a reference to a single mode of transport, represented by a custom implementation of the **AbstractMode** class. These classes, including **BusMode**, **STrainMode**, and **MetroMode**, each represent a specific mode of transport. Upon creation of a mode object, the object creates and subscribes to the output of a series of virtual sensors, represented as implementations of the **AbstractSensor** class. A mode also keeps a reference to the service, allowing it to communicate back to the service when need be.

Each sensor object, once created, has a reference to the mode that created it. The sensor object creates String output in a standardized format and sends this information back to the parent mode every time it updates its sensor information, based on some sensor-specific frequency. The virtual sensor classes have been designed to sense the environmental context described in the previous section, so they include a **WifiSensor**, **CellSensor**, **ActivitySensor**, and a **TimeSensor**.

In the mode object, this sensor input is analyzed and the result put into state variables. After every sensor update has been analyzed, the state variables in the mode are then evaluated. If they meet certain criteria, the mode will send a signal back to the **TrackerService**, asking it to switch to a new mode of transport based on the state variables.

The **TrackerService** then sends a kill signal to the mode which in turn sends a kill signal to each sensor it has subscribed to. These kill signals end any currently running processes in the sensor objects. Then the **TrackerService** replaces the current mode with a new mode object based on the information that the previous mode object sent back. This cycle repeats itself while the service is kept running.

Special modes are used for battery preservation, these include **WaitingMode** and **MovingMode**, both of which only use activity recognition, and do not perform any Wi-Fi scanning.

---

[4] Source code is available at: https://code.google.com/p/public-transportation-tracker/

The widget represents a way for the user to directly manipulate the service by "forcing" certain transport modes in case the context cannot be sensed. This initiates a special **ForcedMode** which only relies on activity recognition to sense a change in current context, but which presents itself to the user as one of the 2 main modes: bus and S-train.

The most common mode is the **DefaultMode**. This is the mode that most other modes default to when exiting. It initiates all available virtual sensors and the only purpose of it is to quickly find another more fitting mode of transport.

User interface

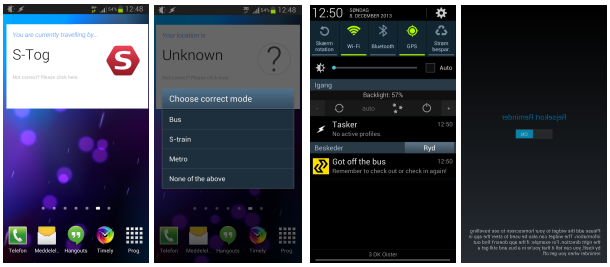

Table 2: App UI

The user interface[5] is simple on purpose. The app is supposed to work automatically in the background, so the only interactive element seen when launching the app by clicking its icon is an on-off switch. Pressing this switch is also the sole required action for the user in order to use the application, as the reminders will show up automatically in most cases.

As an aid, a widget is provided which shows the current detected activity as one of: Unknown, On Foot, (phone is) Sleeping, Bus, S-train, Metro, or (service is) Off. This gives helpful feedback to the user whether the system is working correctly or not. The widget also contains the other interactive element of the application, which is a way to correct the current detected mode of transportation. This is done by clicking the widget and selecting the correct mode of transportation: Bus, S-train, Metro, or None of the above.

The reminders are standard Android notifications that appear as icons in the top bar with a notification message revealed when swiping down. They can be swiped away as any other notification on Android.

[5] Demonstration video: http://www.youtube.com/watch?v=AwHwqoGhI1Q

EVALUATION

Our evaluation consists of multiple methods to answer the following research question:

- **How viable are automatic, environmental context-based reminders as a solution to the check-out issue?**

Introduction to evaluation methods

The system was evaluated using both subjective, qualitative methods and objective, quantitative methods. The subjective evaluation consists of

- Evaluation of S-train and bus behavior.
- Self-evaluation of issues found when using the app in Copenhagen.
- Alpha testing, where users were asked to use the app on their Android phone for 2 days and answer a short questionnaire to document their experiences.

The objective evaluation deals with

- Accuracy of the app based on data from three real world data collection runs in Copenhagen, each 3 hours long. The data measured includes false positives, missing notifications, missing detections, and the delay between getting on a vehicle and detection and the delay between getting off a vehicle and receiving a notification.
- Battery usage tests, which is a comparison of how much battery each mode of transport uses in a lab environment when the service is running compared to when the service is not running.

Evaluation of S-train and bus behavior

Data for how long busses and trains are holding still was collected at Nørreport St. on a Thursday from 9.00 to 9.30, and for S-trains once more from 11.20 to 11.50 at Copenhagen Central Station on the same day.

Busses on average hold still at Nørreport St. for 42 seconds, while S-trains hold still for 33 seconds.

The Copenhagen central station results for S-trains were included because trains typically hold still for much longer there before continuing. We found that S-trains did indeed hold still for much longer, 1 minute and 57 seconds on average. This gives us a lower bound of 33 seconds and upper bound of 1 minute and 57 seconds, where the lower bound gives a sense of how long time must pass before we can confidently say we are on a train. This information is important to prevent false positive reminders.

Analyzing DSB departure information for S-trains, the average time between stops was 2 and a half minutes, while

the shortest average time between stops was ~1 minute and 45 seconds for trains on line F. The average travel time as of 2009 was 22 minutes, which gives the application more than enough time to recognize the S-train [20].

Self-evaluation

Our own field tests were conducted with two android smartphones in the Copenhagen public transport system: a Samsung Galaxy S3 and a Google Nexus 5.

Wi-Fi sensor:

While testing the Wi-Fi in busses and trains we found several issues:

Busses:

- Access point was turned off, eliminating the possibility of recognizing the bus.
- Weak signal from access point, eliminating the possibility of measuring the signal strength as an indication of how close the bus was.
- Change in signal strength. Often the signal would good while the bus was holding still i.e. at a bus stop. When the bus accelerated or was in motion, the signal would drop drastically and sometimes even drop.
- Weaker signal while away from the access point, which is mounted in the front of the bus. Weaker signal was also observed when the bus was completely full of people usually in the rush hour.

Trains:

- Some trains have several access points installed in different spots, which means the signal strength is very different on the two. We observed that the signal was dropped on at least one access point.

Cell sensor:

While testing the cell sensor in the metro we encountered these issues:

- 4G network was penetrating the underground stations with a low signal, making the phone jump from a good 3G network underground while travelling in the metro to a low signal 4G cell tower placed above ground.
- Android reported sometimes reported "unknown" cells while jumping from cell to another cell.

- Data collection (war driving); we only have cell id's from one network provider, the company 3 (Three). If cell id's change due to the company changing their cell towers, it has a direct impact on our apps functionality.

Activity Recognition sensor:

Activity Recognition is an API provided by Google, and is based on their data mining and algorithms, which make them battery efficient and often pretty reliable, depending on the activity recognized.

The On Foot and Still activities were the most accurately recognized activity, however we found that:

- In Vehicle activity was falsely recognized while holding the smartphone in the hand and shaking lightly (experienced while standing outside and freezing) and sometimes while using escalators.
- Metros and S-trains were rarely recognized as In Vehicle, because they ride on tracks and therefore are more still, which makes it hard for the accelerometer to pick up that you are riding the Metro and S-train.
- Busses were mostly recognized as in vehicle. But this depended on the vibrations of the bus.
- Placing the phone directly in contact with a bus or metro increased the likelihood of the sensor recognizing in vehicle.

Modes:

We found modes to be hard to define, to be working in every situation.

We found that we needed a tolerance in case of Wi-Fi signals were lost and refound while travelling in busses or trains.

Furthermore we found that the "on foot" activity recognition is a good choice for several uses when tracking transitions between modes.

Alpha-testing evaluation

Seven people were asked to use the app on their Android phones when travelling with public transport in Copenhagen. After two days of usage, the users gave feedback on the usability, functionality, and accuracy of the system by answering a short questionnaire.
[19].

Five of the seven users tested used Rejsekort, the rest using regular tickets or ten-trip cards. Four out of the five Rejsekort users had experienced getting a fine for forgetting to check out. All users were positive of the concept behind the app, but had some remarks regarding current errors in the system. None of the users had tried the existing check out apps.

User 1 was male and 28 years old. While using the app, he had no false positives and no missing reminders and was mainly travelling on the S-train. User 1 thought the app's interface was simple, but could give more user feedback.

User 2 was female and 27 years old. She primarily used the app in busses and had issues with receiving reminders while still on the bus in one case. The app also did not register her getting off the bus in another case, which made her receive the reminder the day after she got off the bus.

User 3 was female and 24 years old. She primarily used the metro to get around and received reminders quickly at appropriate times. She did at one point correct the mode using the widget when getting on the metro, though.

User 4 was male and 29 years old. He used all three modes of transportation to get around. The app gave him reminders in some cases, but he had one missing reminder when getting off the metro and one when getting off the bus. He thought the app was a good proof of concept and with potential.

User 5 was male 58 years old. He did get a correct reminder while getting off the bus. _He would like to be reminded in the bus. He thought the widget needed an indication of whether the service was running or not.

User 6 was female 62 years old. Didn't get reminders in the metro because of her cellular network being TDC. She experienced four false reminders while travelling in a car, but didn't notice any busses nearby while driving. Se got one premature reminder in the S-train, just before getting off the train, but the two other S-train trips gave her correct reminders. When testing in bus 1A, she didn't get a reminder, most likely due to no Wi-Fi.

User 7 was male and 26 years old. He didn't have a Rejsekort. He tested the app in two S-trains and got reminders both times, the first time just while passing the check out terminals. He thought the widget was easy to use and I would like to use the app in the future and had not other problems with the app.

Accuracy evaluation

Here are the results of registration and notification delay compared to the ground truth. The results were collected on a Google Nexus 5 phone.

	Sum	Average delay	
Total cases	30		
Bus registrations	3	47,00	seconds
Metro registrations	6	**16,33**	seconds
S-train registrations	10	*134,00*	seconds
Bus notifications	3	47,00	seconds
Forced (bus) notifications	6	43,50	seconds
Metro notifications	7	**40,14**	seconds
S-train notifications	10	*55,20*	seconds
All registration delays	19	96	seconds
All notification delays	26	47,50	seconds
	Sum	**Ratio**	
Premature notifications	1	3,33%	of total cases
False positives	1	3,33%	of total cases
Notification missing	2	6,67%	of total cases
Bus mode needed to be forced	6	66,67%	of total bus cases
Metro missed notification	2	22,22%	of total metro cases

Table 2: Accuracy evaluation

Evaluating the accuracy of the system has proven time-consuming. Three sessions of around 3 hours each were spent measuring the delay of registrations and notifications of the app as compared to the ground truth. These sessions were spent doing random travels with public transport throughout Copenhagen. We recorded 30 cases in total of either a registration, a notification, or both.

In these sessions, we found that the app did not correctly show a notification twice, or in slightly more than 22% of the cases, when travelling with the metro. There were notifications present in all cases recorded for busses and S-trains.

However, the app did not correctly register the bus mode of transport in two thirds of the recorded cases. This was used as an opportunity to test the ability of the user to force a specific mode (bus) and thereby test the how fast notifications materialised in this way. Notifications in forced mode were faster than both traditional S-train and bus notifications.

The metro mode had the lowest delay when both registering and notifying the user at 16,33 and 40,14 seconds, respectively. The slowest was the S-train mode at 134 and 55,2 seconds respectively. The average notification delay for all recorded cases was 47,5 seconds while it took 96 seconds on average to register..

During our sessions we recorded a single false positive (a bus) and a single premature notification (in an S-train).

Battery tests

For this lab test, we tried testing our most intensive mode in our app, the mode with all sensors turned on, and a test with the app not running. The Android operating system has several features to enhance battery life, we noticed i.e. processor cores were brought down in speed and the system has a deep sleep mode which is very battery efficient.
We conducted the test by leaving a Google Nexus 5, fully charged (2300 mAh battery), on without mobile data or wifi data turned on for 3 hours, the approximately consumed battery was 34 mAh.

Hereafter we conducted the same test, this time we launched our app, and programmatically turned on our Default mode with all sensors present and left it for 3 hours. This test showed a decrease in battery life of approximately 46 mAh. The difference between the first measurement without the app and the one with the app on was: approximately ~12 mAh in 3 hours.

We tried a last test with the same parameters, testing our Default mode again. This time we tested for 8 hours and the results that came were exactly the same as for the 3 hour

test with a decrease of 46 mAh. This let us to believe that Android has some sort of role to play in the results and we do not rely on these too much. Battery in general is very hard to measure, hence the battery itself might be over or undercharged even though the operating systems sees it as a fully charged 2300 mAh battery. We are not entirely sure why these tests came back with these relatively small differences, but we haven't noticed our two phones plummet in terms of battery life while testing the app over the course of the last month.

DISCUSSION

Comparing the train behaviour data with the detection times, we found that the current rates of detection for S-trains of around 2 minutes is consistent with the approximate average time between stops of 2:30 minutes. Line F might pose a problem for detecting the S-train if the users only travels one stop, but we find this to be a rather rare case. Ideally the detection of S-trains should be brought down another 15 or 30 seconds. However, this would cause more false positives on average as this is effectively lowering the confidence of the detection result.

The average time holding still for busses on Nørreport St. is 42 seconds while average bus notification time is ~40 seconds. If bus notification times can be brought down another 10-15 seconds on average, this might even allow the user to go back check out at the same bus he just came out of. However, faster notifications would mean effectively lowering the confidence of the result and will cause more premature notifications.

The results of the accuracy evaluation show us that the ability to force a specific mode is absolutely essential for this type of system to work in the Copenhagen busses. We speculate, backed by field measurements during our tests, that missing or weak Wi-Fi signals are the main cause of these missing registrations and thereby potentially missed notifications.

Another cause for concern is the missing notifications in the metro mode. While this mode was the closest to the ground truth in terms of its delay, the mode switching logic will need to be reworked slightly for it to work properly for everyday usage. We speculate that a more thorough and careful scanning of cell signals in the tunnel will help a great deal in solving this issue.

The average notification time at 47,5 seconds we deem good enough for everyday usage, although adjusting the frequency of the various sensor outputs will most likely bring this delay even further down. The average registration delay of 96 seconds is not a key concern as this is still well below what a typical trip with public transport takes, however for short one-stop trips it could make the service too slow to react.

The alpha testing showcases the existing problems with the system when it comes to accurately delivering reminders in certain situations, mainly when leaving busses. Overall, users were positive of the system's potential and its user interface, but the feedback also reveals that issues with false positives and missed reminders might be more prevalent on certain Android devices than the ones used for our self-evaluation and the objective tests. Especially false positives in terms of false bus reminders, ie. while driving a car is of a concern. Some problems occurred in certain atypical situations where the current rule-based logic did not guess the right context.

The battery tests were too short to conclusively prove anything, although the low recorded battery usage of the service could be used an indication that the service is not particularly battery-intensive.

CONCLUSION

After developing and testing a prototype system limited to Copenhagen, we found that the concept of automatic check-out reminders resonated with our alpha-test users and that the reminders show up somewhat consistently in our own tests - with the exception of those in the metro. However, there were certain situations where the prototype did not deliver a satisfactory experience, primarily when users experienced delayed or missing reminders - or when users received reminders while still on the vehicle.

Some of these errors can be attributed to internal deficiencies, such as an the incomplete switching logic in the AbstractMode implementations and an incomplete collection of environmental context data; for example, cell IDs in the metro tunnel. These types of errors can be corrected in future versions of the software with adequate field-testing and more data collection.

External issues, mainly related to differences in hardware sensors on user devices, fluctuating Wi-Fi signal properties, and accidental proximity to busses or trains, are much harder to correct. It is possible to lower the frequency of premature or missing reminders through carefully evaluating special situations and testing with many different devices in different situations. Unfortunately, these issues cannot be totally eradicated as the system will always have to make a guess based on typical user behavior and cannot reliably detect a large number of special situations, where user behavior differs.

Overall, we found the concept of automatic check-out reminders to be a viable alternative to the existing category of apps that attempt to solve the check-out issue. The system can be made to work well within the boundary that we define as typical user behavior, but it does not work as well in atypical situations.

REFERENCES

1. Trafikudvalget, Folketinget, Status for udrulning af Rejsekort i Danmark, 2010-11 (1. samling) TRU Alm.del Bilag 272 http://www.ft.dk/samling/20101/almdel/tru/bilag/272/983343/index.htm
2. Fagbladet Ingeniøren, Krøyer, K., Krypteringen i Danmarks nye Rejsekort er allerede knækket, 30-05-2008 http://ing.dk/artikel/krypteringen-i-danmarks-nye-Rejsekort-er-allerede-knaekket-88641
3. Dagbladet Politiken, Guldagger M., Rejsekortet æder rabatten til studerende og pensionister http://politiken.dk/forbrugogliv/livsstil/dintransport/ECE1991683/Rejsekortet-aeder-rabatten-til-studerende-og-pensionister/
4. DR Nyheder, Rejsekortet er på kant med loven, 24-05-2013 http://www.dr.dk/Nyheder/Indland/2013/05/24/225440.htm
5. DR Nyheder,Nørtoft M., Rejsekortet: Her er de seks største kritikpunkter, 20-02-2013 http://www.dr.dk/Nyheder/Indland/2013/02/20/141252.htm
6. Version2, Meister M., Glemmer du også at tjekke ud? Rejsekort gav 200.000 dummebøder, 19-02-2013 http://www.version2.dk/artikel/opgoerelse-200000-glemte-tjekke-ud-med-Rejsekortet-i-2012-50500
7. DR Nyheder, Vestereng C., Mobile apps klarer Rejsekort-problem, 18-06-2013 http://www.dr.dk/Nyheder/Indland/2013/06/18/134809.htm
8. Android Developers Blog, Hartrell et al, Social Gaming, Location, and More in Google Play Services, 16-05-2013 http://android-developers.blogspot.dk/2013/05/social-gaming-location-and-more-in.html
9. Version2, Strøm, Georg, Det skaber nye fejl at rette fejl i rejsekortet, 01-12-2013 http://ing.dk/artikel/rejsekort-direktor-vi-troede-vi-kunne-lose-alt-med-et-big-bang-103544
10. Zhang, M. and Sawchuk A. Motion primitive-based human activity recognition using a bag-of-features approach, Proceedings of the 2nd ACM Sighit International Health Informatics Symposium,2012 http://sensorlab.cs.dartmouth.edu/urbansensing/papers/urbansense08_proceedings.pdf#page=52
11. Muthukrishnan, k., Lijding, M. and Meratnia, N. Sensing motion using spectral and spatial analysis of WLAN RSSI. Smart Sensing and Context, 62-76 , 2007, Springer Berlin Heidelberg
12. Transportministeriet, bilag, 06-04-2011 http://subsite.kk.dk/~/media/B315F8DC523F4061B142543C74D90B33.ashx
13. Fagbladet Ingeninøren, Bredsdorff, M., Samlet regning for Rejsekort: Fem milliarder kroner, 09-07-2012 http://ing.dk/artikel/samlet-regning-Rejsekort-fem-milliarder-kroner-130633
14. DR Nyheder, Sørensen, L. M., TIDSLINJE: Rejsekortets historie - en milliard på 14 år undervejs, 19-06-2013 http://www.dr.dk/Nyheder/Indland/2013/06/18/153434.htm
15. Octopus card, Operation of the Octopus Card in Hong Kong, 2007 http://www.legco.gov.hk/yr06-07/english/sec/library/0607in08-e.pdf
16. World Bank & ADB, Cities at a Crossroads: Unlocking the Potential for Green Urban transport, 2012 http://siteresources.worldbank.org/EXTINFRA/Resources/UrbanMassTransport_Web.pdf
17. Reddit, Nogen interesse i en app der automatisk kan se når du rejser i offentlig transport og hjælpe dig med det?, 20-09-2013 http://www.reddit.com/r/Denmark/comments/1mrok6/nogen_interesse_i_en_app_der_automatisk_kan_se/
18. Interaction Magazine, Meiken Hansen et al, Resjekort put to the test, 11-10-2011
19. Appendix, Alpha Testing with end users, December 2013 https://docs.google.com/document/d/1t8PgcYnQfRLGlSsxp3NQ8Wcd-DIJjXFxLdKXbhLH7vk/edit?usp=sharing
20. Commute Media, Commute TV - Passagerprofil, 2009 http://www.commutemedia.com/media/11691/s-tog%20passagerprofil.pdf
21. Martin, E., Vinyals, O., Friedland, G. and Bajcsy R. Precise Indoor Localization Using Smart Phones, MM '10 Proceedings of the international conference on Multimedia, 787-790, 2010, New York
22. Herrmann, R., Zappi, P. and Rosing, T. Context Aware Power Management of Mobile Systems for Sensing Applications, Mobile Sensing, 2012, Microsoft Research
23. Chen, Y.C., Chiang, J.R., Chu, H. and Huang, P. Sensor-assisted wi-fi indoor location system for adapting to environmental dynamics. MSWiM '05 Proceedings of the 8th ACM international symposium on Modeling, analysis and simulation of wireless and mobile systems. 118-125, 2005, ACM New York.