Efficient Event Detection in Public Transport Tracking

Ármin Petkovics and Károly Farkas
Department of Networked Systems and Services, Budapest University of Technology and Economics, Hungary
petkovics@hit.bme.hu, farkask@hit.bme.hu

***Abstract*— Personal mobile devices are widespread and carried by their users most of the time over the day. Thanks to the integrated sensors they can report about visited places, movement types and speed of the users. However, efficient stopping event detection on public transport vehicles is still a challenge. These events, associated with the coordinates of real stations, can be useful to update public transit timetables according to real-time traffic. In field tests we evaluated the most commonly available suitable sensors' precision and efficiency and developed our Stopping Event Detection Algorithm (SEDA), which utilizes only the accelerometer to find potential stopping times and the Wi-Fi sensor to validate or discard them by a novel localization method. Wi-Fi is used only 6.66% of the time of actual traveling on public vehicles. Our algorithm is shown to recognize properly 82.9-89.47% of public traffic stations while consuming daily only 13% the capacity of an average smartphone's battery.**

***Keywords* — Mobile sensing, event detection, crowdsensing, energy efficiency**

I. INTRODUCTION

In bigger cities traffic can be unpredictable so most public transit lines that are influenced by traffic can bear shorter or longer delays. Also, some weather conditions like rainstorms or an extreme snowfall can hardly influence the traveling speed of public and private vehicles in the city. To the passengers it is even more important not to wait too long at the stops in case of extreme weather conditions. This creates a challenge that can be addressed with a system that monitors all the public transport vehicles and constantly updates their timetables and makes them available within an online service where the users can (geo-dependently) check the "real" arrival times at the current stop of any traffic line that operates there. This system can gather the data needed in different ways.

The most obvious solution is the deployment of sensors along the traffic routes or on the vehicles. Their usage inheres capital expenditures at the beginning and physical deployment of each sensor. Besides buying and deploying the sensors there are also expenditures and challenges with enabling each sensor to communicate with a central server and maintenance costs are also applying. Although today it is not the cheapest and the most efficient method, its usage in public transport is well elaborated and also patented in the US [1].

A novel approach is the usage of sensors that are somehow attached to human body. They represent a variant of mobile sensor networks which rely on people's smartphones as they utilize the sensors integrated in these devices. Smartphones are uncontrolled mobile sensors as their mobility is not restricted as of those sensors which are deployed on public vehicles. They move along with their owner and collect the information about speed, acceleration, connected cell towers, Wi-Fi hotspots in sight, etc. The way they provide information can be divided into two categories, participatory or opportunistic, whether the owner plays an active role in sensing or not. In participatory sensing the user helps the sensing process with manual intervention indicating when he/she gets on a public vehicle, name the traffic line, and the stop where it happened. At automated or opportunistic sensing the user plays a passive role in the process as the sensing tasks are carried out automatically by an application running in the background. Although participatory sensing can lead to more exact data, opportunistic sensing is more reliable because it lacks the human factor and can operate providing constant data feed.

The organized form of mobile sensing is a promising way for real-time data harvesting with the help of a big group of moving sensing users and it is called crowdsensing. It does not need physical deployment so expenses are much lower than in ordinary sensor networks (it utilizes the crowd's smartphone-sensors). The sensing application collects and proceeds data to a server. The processing could be done either on the user side or at the server. If used in real-time applications, users of the crowd should have mobile Internet or Wi-Fi access. Sending big amount of data through cellular network causes higher expenses and energy consumption, so the larger part of processing should be done at the user side.

The paramount challenge that compromises users to join crowdsensing is the battery life of smartphones which can be affected badly by all-day-long tasks which are utilizing several energy-hungry sensors. GPS is known to be one of the most energy-demanding smartphone sensors, but it used to be the primary source of location information.

We address consumption reduction of user devices collaborating in public transport tracking crowdsensing tasks, with a complex energy efficient method. Stopping Event Detection Algorithm (SEDA) aims to differentiate between the states of traveling on public transit vehicles, which includes potential stopping event detection (SEDA 1) solely relying on raw accelerometer data, and evaluation of the findings, eliminating false positive stops altogether with localizing them (SEDA 2) so the sensed stopping times could be compared to those in the fix public schedule, so the crowdsensing application can update the timetable accordingly and help other passengers avoiding unnecessary waiting for late rides.

Since the localization task is cumbersome in dense urban areas where public transport tracking mostly needed, we evaluated the accuracy of several well-known strategies and proposed a novel solution that is more trustworthy, which lacks fluctuations in precision and which shows lower consumption values than known methods.

The targeted tasks are fundamental parts of public transit-tracking, so the improvements achieved in their solution by our approach can be applied in all related scenarios.

978-1-4799-3200-9/14/$31.00 ©2014 IEEE

The remainder of the paper is organized in the following way: state-of-the-art is discussed in Section II, our approach and solution to the problem is presented in Section III, Section IV summarizes results and efficiency values, while section V concludes the paper.

II. RELATED WORK

In mobile sensing applications energy consumption is not primarily important because of CO_2 emission but to prolong battery life of the mobile devices. As most crowdsensing applications, event detection in public transit tracking also includes localization tasks to determine where the sensed data is originated from. Among the relatively high number of localization techniques only a few consider the energy consumption impact of their method. Event detection makes precise activity detection necessary which primarily relies on accelerometer readings and is often supported by additional sensors.

Bulut and Demirbas in [2] give a method for *mobile localization with varying accuracy levels depending on the devices' type of movement,* speed and distance to the points of interest where the most accurate location is needed. They distinguish idle, driving and walking modes. Based on these the system chooses the needed location accuracy and accordingly the needed sensors (GPS, Wi-Fi, cellular localization) and their sampling intervals.

Thiagarajan *et al.* present an *activity dependent localization technique* aimed for public transport observing purposes [3]. To more precise sensor calibration it differentiates between sleeping, stationary, walking, traveling by car and traveling on public transport vehicle. The accelerometer query frequency rises for movement types in the same sequence they are listed above, and additional sensors are activated while walking (Wi-Fi + cellular) and while traveling (GPS). They also do *route identification* where they use the least squares optimization algorithm which finds the closest fit among known bus lines for a sequence of sensed GPS samples.

Wang *et al.* in [4] present also an *activity dependent mobile sensing framework* where the novelty is that sensors' duty cycles (sampling frequencies) are determined heuristically from earlier observed data, so changes happen on the fly when the user changes his/her movement type. They also use GPS at some cases and microphone readings are involved to their system as well.

Cardone *et al.* present McSense [5], *a geo-social model driven crowdsourcing platform which improves task assignments* by collecting information about its users. They are grouped by geographical regions, type, CPU and sensors of the phone they are using, usual paths they are moving by and their battery level is also considered before they receive a task to work on.

Liu *et al.* in [6] deal with the task of maintaining the *quality of information* collected by the crowd. They determine the adequate number of sensed data packages that are enough for statistical processing in the concrete situation. They introduce fairness by *minimizing the variance in energy consumption between the crowd's users* with the extended Gur Game. Efficient *route identification* for public traffic lines is investigated by Zhou *et al.* in [7]. They take into account only the cell-tower IDs seen during the trip and claim that absolute localization is unneeded for route discovery. They sequence-match among received and earlier saved cell-tower ID sequences to determine the traffic route by a modified Smith-Waterman algorithm. The energy cost of collecting cell-tower IDs is negligible because mobile phones always monitor the available cells around to choose the best signal-to-noise ratio.

The rich work of this field solves several problems that a crowdsensing project could face: maintaining quality of sensed data, better task-assignments, traffic route identification, activity dependent localization, etc. Authors solve the task of distinguishing between traveling and other movement types but stopping event detection in public transportation is not addressed yet. We solve this unelaborated task together with efficient localization of the found stopping places without relying on GPS or cellular localization, but utilizing Wi-Fi scanner for very short periods of time during the sensing users' public transit travel.

III. STOPPING EVENT DETECTION

Stopping event detection aims to differentiate between the accelerometer readings of a user on a moving and a resting public vehicle. At public transport tracking the most valuable information is the timestamp when a public vehicle stops at a station, because here the system can compare the sensed time with the fix timetable. After the potential stopping events are found, they have to be analyzed and localized to link each one of them to the physical place where the stopping event occurred. Our detection method requires the location information, because the place where the potential stopping event was sensed decides whether it was a real station or somewhere in between stations, e.g., at a traffic light.

A sensing application was developed for data collection from all the possible sensors the Android API allows. Tests were carried out with different Android smartphone models, through a 3 month period of time including high- and low-traffic hours, clear and cloudy weather on and near the Budapest tram line 4 and 6 which are the city's public traffic arteries. During data collection, testers were manually indicating the public transport stops as the baseline for further comparisons. The aim with inner-city data collection was to develop and test our algorithms on real-life data. The accuracy of all the subtasks were evaluated continuously.

SEDA consists of two parts: SEDA 1 finds potential stopping events from accelerometer readings, while SEDA 2 eliminates those stopping events - by using precise localization - which happened outside the real stops (stopping at traffic lights or due to unexpected traffic situations).

A. Detecting potential stopping events

This task is not straightforward, because when the tram stops, the traveling user is still moving the sensing device in some way, depending on where the phone is kept on his/her body and which position is the user in.

Accelerometer is the primary source of information for activity detection, which shows us the momentary force that is affecting the smartphone along the three axes. Following the general approach we calculate the root of sum of squares of

the three values to find absolute acceleration which is regardless to the phone's orientation. Accelerometer readings do not describe the speed of movement. Though there are some attempts to estimate the possible moving speed [8] and path traveled from accelerometer readings, their accuracy decreases dramatically with time.

Basic activity detection is well examined in the literature and we also implemented a simple algorithm that separates resting, walking and traveling modes on the fly according to the average acceleration levels and their dispersion.

However, detecting the potential stopping event is more complicated. First of all, we noticed that different devices use different intervals for sending sensor data through the API. For easier and fair comparison of the results from different devices' datasets we rescaled their readings to 200ms intervals as a first step for enabling correct sensed data processing.

Our algorithm differentiates crowdsensing users who are standing or sitting during public transit travel. These two cases behave differently as there is higher noise level in accelerometer readings of standing users. It is possible to define threshold values that are applicable for both situations, but that would result in much more false positive stops (stops outside stations due to traffic lights or noisy accelerometer data) sensed in SEDA 1 which would badly influence the overall consumption values of the SEDA 1 and 2. Standing and sitting are easily separable by known algorithms with an accelerometer, like the method described in [9] or by making the sensing application adaptive to the average noise level.

SEDA 1 algorithm:

1. Smooth out accelerometer readings by a 3-second long moving average.
2. Give first approximation of stops with threshold values of 0.1 m/s^2 for sitting or 0.25 m/s^2 for standing users.
3. Apply our low-pass filter for eliminating glitches.
4. Quantize filtered results of step 3, with thresholds 0.5 for sitting and 0.65 for standing users. Unity steps in the result denote the potential stopping times.

The filter of step 3 was designed using the *'fdatool'* toolbox for Matlab. The order of the FIR filter was set to 124 samples, with a cut-off frequency of 100 Hz. The filter's order implicates the maximum length of a glitch which it eliminates.

Figure 1 illustrates application of SEDA 1 on a 3-minute long dataset showing step-by-step results for better understanding. Result of the second step (marked red) shows that for three very short periods, the moving average of step 1 (blue line) goes below the threshold level when the tram was actually traveling, resulting in glitches. To eliminate them we apply our low-pass filter (step 3): the dashed green line's unity steps (value changing from 0 to 1) denote the places of SEDA 1's findings: the potential stopping times. As we look at the two small black circles in the lower part of the figure which indicate manually sensed stopping events, we can see that SEDA 1 found the stops exactly with negligible, few seconds slip in time.

SEDA 1 algorithm can be evaluated by the proportion of false positive (junk sensed stops in situations like traffic jams) and false negative potential stops (stops that are missed by SEDA 1 due to noisy accelerometer readings) to the number of real stops. While false positive stops are further evaluated and corrected by SEDA 2 (see below), false negative stops immediately result in losing a stop. Thus we determined threshold values rather to result more false positives and accordingly, less false negative stops. Accuracy values for both SEDA 1 and 2 are presented in section IV.

B. Localization

Stopping event detection needs accurate localization so the algorithm can decide which stopping events - sensed by the accelerometer - happened truly at a station, and which stopping events could be linked to traffic lights or other traffic situations. Our aim is to find only those stops which happened at the known stations.

Device manufacturers and operating system developers already take this task seriously. While older Android versions offer only two different settings how to obtain location (GPS and Network (which means cellular + Wi-Fi) localization), the latest Android 4.4 (Kitkat) system offers three types: high

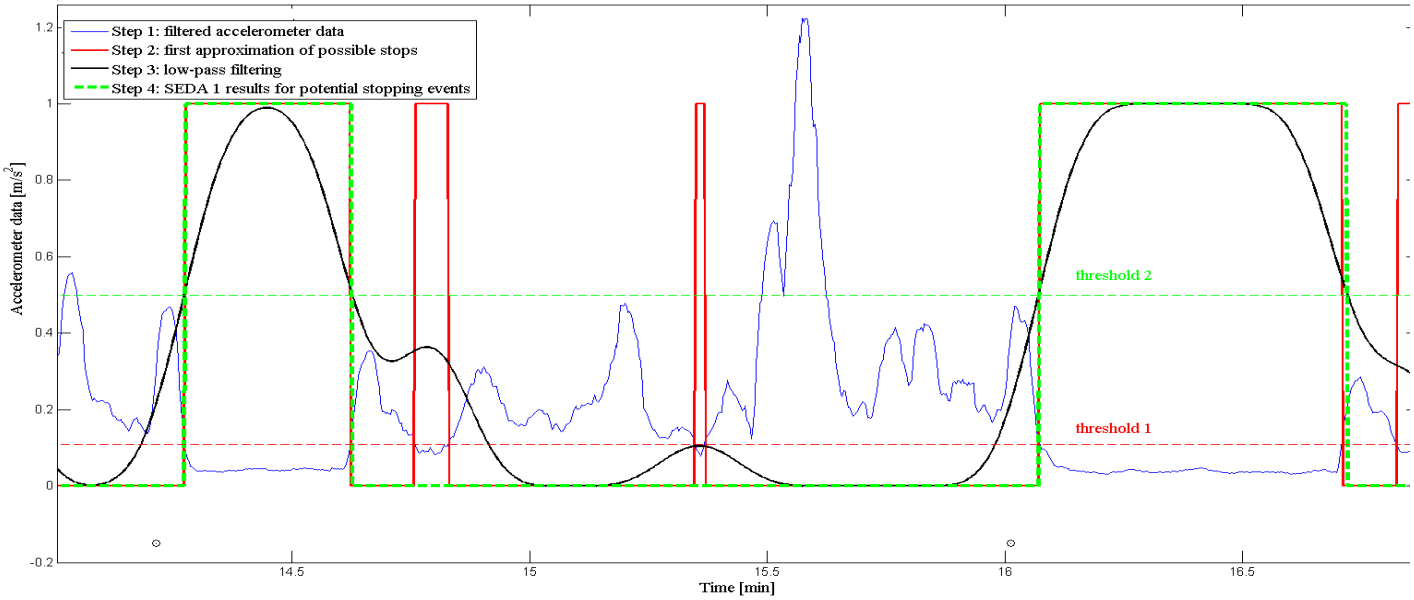

Fig. 1. A 3 minute accelerometer reading sample with the 4 steps of SEDA 1. The time of the two real stop events are depicted at the bottom with a circle. (device Nexus4, sitting sensing user)

accuracy (which means the combination of GPS, cellular, Wi-Fi and other possibly helpful sensors in this task), "battery saving" mode (Wi-Fi + cellular) and "device only" which means the use of GPS only. The first two modes use Google location services to estimate location faster and more accurately while the third option excludes Google from the localization (location info determined by GPS is anonymous). Also for each installed application the user can decide whether it receives location information and if it does how accurate localization is available to it [10].

Most mobile operating systems are taking care of location sensing problems with high priority to make them more efficient and more accurate but these solutions are mostly supported on the latest devices, with the newest operating system versions installed. Older devices rarely get the updated operating systems. Therefore the needed sensors and accuracy classes for SEDA are still to be defined manually to achieve the lowest possible energy consumption across all versions of Android operating systems.

Our primary goal was to solve this task in the simplest way possible (avoiding computing intensive tasks which lead to unnecessary energy consumption of the devices) with only low-power consumption sensors involved. First of all, we tested how the different known localization methods deal with this problem in dense urban areas of Budapest.

We had high expectations for *cellular localization* because high buildings in the city result in a lot of users and accordingly high base station density. This is expected to lead to frequent handovers at traveling user devices, which could mean that knowing the position of the base station the sensing user is connected describes the user's position with very good approximation. During field tests we collected all the cellular coordinates the devices sensed during the trip. Figure 2.a visualizes a particular dataset on the map, which shows that in case of a particular round trip with tram 6, network localization results followed the Grand Boulevard with good approximation. But, when we checked the numerical values and variance of localization accuracy by calculating the difference between real and cellular coordinates at each stop we found results depicted on Figure 2.b. One can notice that there are sometimes higher errors at some stops. These errors are due to that some parts of the trams' path are straight enough that the mobile phone can stay connected to the base station that is near the previous stop or maybe the stop before that. Handovers occur when the mobile node leaves the coverage area of the previous base station, or the signal-to-noise-ratio (SNR) of another base station is better than the previous'. But, if the next cells are crowded the handover algorithm can chose to stay connected to the previous base station. This happened several times in our tests, e.g., in Figure 2.b device Galaxy S2 had no handovers between "32esek" and "király" stops (the distance between the measured cellular location and the real location rises linearly).

Summarizing all results of **cellular localization** we calculated an **average error of 183.33 meters.** Typical average values of per measurement accuracy varied from 93.47 to 272.85 meters. On average this is not a bad result because tram stops are far away from each other from 350 up to almost 7-800 meters, but there are some late handover scenarios when the error is too high. We concluded that this localization method could not be used alone.

We visualized and calculated the accuracy levels of ***GPS localization*** of several tests in Figures 3.a and 3.b. After we summarized all the test results the **average error of GPS localization was 163.19 meters**. Average per measurement values varied from 21.74 to 646.01 meters depending on the type of smartphone / weather conditions. Comparing the results of cellular and GPS positioning we can conclude that GPS is on average only 12.34% more accurate while its consumption is around 2.5 times higher [2] than cellular localization's.

Neither GPS nor cellular localization gives us in every

Fig. 2.a Accuracy of cellular localization (device: Nexus 4).

Fig. 3.a Accuracy of GPS localization (device: Nexus 4).

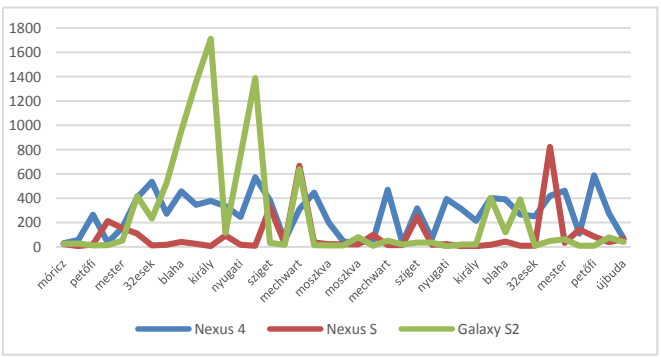

Fig. 2.b Accuracy of cellular localization in meters on 3 different devices, two way trip of tram 6.

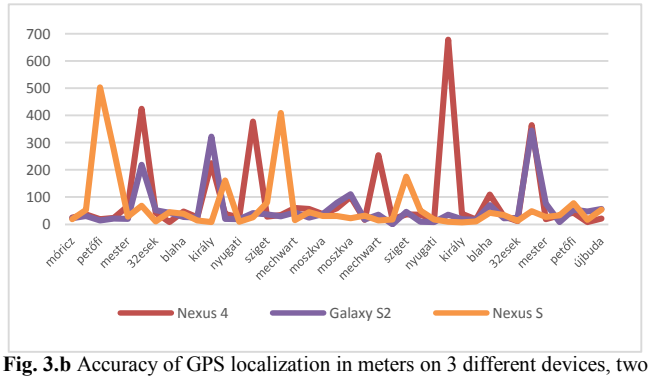

Fig. 3.b Accuracy of GPS localization in meters on 3 different devices, two way trip of tram 6.

situation adequate results with accuracy levels that we can rely on exclusively without additional help from other sensors. Our initial expectations that GPS can be left out from our localization technique was shown to be true, because similarly accurate results come from cellular localization. Anyways, a new localization method should be found to filter out false values from SEDA 1.

The idea of ***Wi-Fi localization*** relies on the fact that Wi-Fi hotspot coverage values goes up to 50-100 meters so sensing the same access points from two different tram stations is almost impossible. Here we came up with the idea of building a **Wi-Fi database** along the two traffic lines we were dealing with by saving the visible hotspots at every station according to several field test results carried out 1 month after one another. Possible problem could be that nobody from the collected hotspots' owners is forced to keep their Wi-Fi network alive, so our Wi-Fi database should be updated periodically. Luckily, in dense urban areas that we are working with there are a bunch of hotspots at every step so our algorithm still be operational if some of them are lost.

Wi-Fi based localization is very efficient because this sensor has to be turned on only when the stopping event detection algorithm detects a possible stopping place: in 4 seconds it finds the list of visible hotspots and it can go offline again. We present SEDA 2 method for filtering out false potential stops found by SEDA 1 and making sensed stops geo-dependent by exact and efficient localization.

SEDA 2 algorithm:

1. Build/update Wi-Fi database at traffic line's stops. Check all neighboring stops for matching hotspots: delete them from both lists.
2. For every potential stopping event: turn on Wi-Fi for 4 seconds, collect visible hotspots. Match the found stops with the database and fill up the Wi-Fi matching matrix.
3. Discard potential stops whose AP's match more than one real stop's AP's.
4. When more (consecutive) potential stops match the same real stops' APs then choose the one with the most hotspots matching as the real stop. Discard the others.

Rule number 4 means that we decide about the correctness of a potential stopping event just after we found a next potential stopping event with hotspots matching the next real stop's hotspots in the database. This can cause a 1-2 minute delay in the decision, but the timestamp of the newly judged stop is still comparable with the fix timetable to predict arrival times for all the coming stations accordingly.

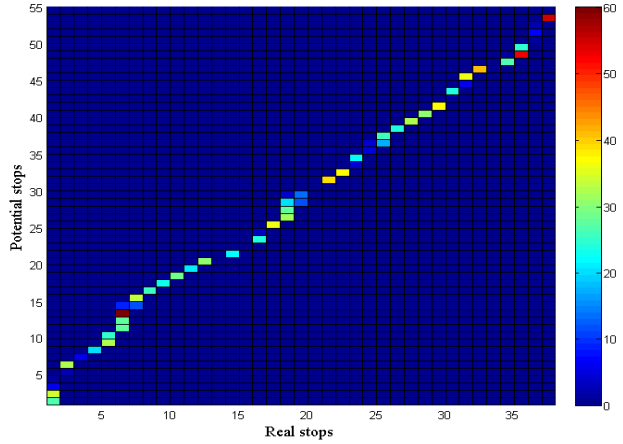

Fig. 4. Wi-Fi matching matrix. Dataset of 37 consecutive stops on a round-trip with tram 6. Device: Nexus 4, sitting sensing user.

Figure 4 depicts the Wi-Fi matching matrix between the hotspots sensed at potential stopping places and those at real stops earlier collected to the Wi-Fi database. Number of matching hotspots are marked by different colors (see label on the right side). The matching matrix describes how SEDA 2 operates: potential stopping events of SEDA 1 (y-axis) are compared with real stops (x-axis) based on the number of matching Wi-Fi APs. Some examples for better understanding: The 7th real stops' hotspots are sensed in four consecutive potential stops (11-14th), among them the 13th has the most matches with the real stop, which is marked by the darker color in the matrix-representation in the figure.

The APs at 28-29th potential stops match the APs of the 18th and 19th real stops which means that both of the potential stops are discarded because in step 1 of SEDA 2 we eliminated the overlapping hotspots, so these ones have to be somewhere in between the two real traffic stations.

An empty column on Figure 4, like the 33th, means that real stop no. 33 was missed by SEDA 1 (it is a false negative).

IV. Results and Efficiency

The accuracy of SEDA 1 could be evaluated alone by calculating the proportion of false positive and false negative stopping events to the real ones. As discussed, threshold levels are set in SEDA 1 rather to find more false positive stops than false negatives, because the latter means that we have certainly lost a stop, but false positives could be filtered out without any stops getting lost during the sensing process.

One can see that in a particular sensing case in Figure 5, where there were 38.74% more potential stopping events (false positives) than real stops, after executing SEDA 2, only 1 false positive stopping time left (at around 12.5 min on the x axis) which increases the final error rate with 1/18=5.56 %. Meanwhile the 3 false negatives (3 missed stations) cannot be filtered out (there are no stopping events at the missed timestamps that can be evaluated with Wi-Fi localization), and they add 3/18=16.67% to the final error rate.

After processing all the datasets from different smartphones that were collected through 3 month of time in different traffic

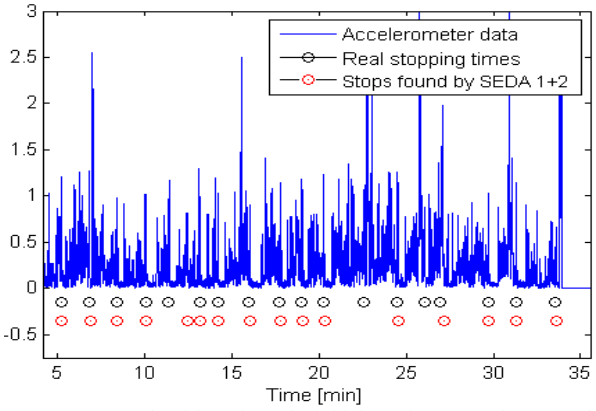

Fig. 5. : SEDA 1+2 algorithms from the initial accelerometer dataset to the found stopping events described by estimated stopping times below the dataset. Real stopping times marked for comparison. Device: Nexus4, sensing position: sitting.

Table. 1. False positive and false negative possible stopping events in both traveling positions and the overall accuracy of SEDA algorithms

Traveling position	False positives SEDA 1.	False negatives SEDA 1.	Overall accuracy SEDA 1+2	Average slip in sensed arrival times
Sitting	44.74%	7.89%	89.47%	10.07 sec
Standing	60.53%	12.2%	82,90%	15.64 sec
Average	52.635%	10.045%	**86.185%**	**12.855 sec**

and weather conditions at different times of the day we calculated the average accuracy levels seen in Table 1.

In the Budapest case of tram line 6, the vehicle travels along its 19-stations path for 29 minutes, which means that in average traveling between two stops and the stopping time last together 91.6 seconds. The percentages from Table 1 (82.9-89.47%) mean that we properly detect around 8 or 9 from 10 stopping places.

A user who is participating in public traffic-related crowdsensing **using SEDA**, **sends a status update of the tram's current position on average in every** 91.6 sec / 86.185% = **106.28 seconds** of the trip. Talking about one lonely user, this is a great number of sensing sent automatically, without any interaction needed from the mobile crowdsensing user.

Calculating with the average time of 91.6 seconds per each station (travelling and resting) and 4-seconds Wi-Fi operation per each potential stopping event, where the average number of potential stopping events is 52.635% (false positives of SEDA 1) higher of the real stops (assuming that half the sensing users are travelling in standing and half in sitting position) we get a utilization percentage of 1.52635 * 4sec / 91.6sec = 6.66% for Wi-Fi scanning.

Consumption values of mobile sensors are similar for different devices, showing a slight decrease in newer models due to further optimization. Earlier researches reported consumption values of 1.37 Watts for Wi-Fi scanning, 0.05 Watts for continuous accelerometer usage with maximal sampling frequency, and 0.32 Watts for continuous GPS usage [11].

Supposing that an average user is awake 17 hours a day and is in steady state at least 5 more hours (at his/her workplace), results a maximum of 12 hours of moving activity per day from which maximally 4 hours are spent on public vehicles (if talking about a big city). **Applying SEDA 1 and 2 on this average daily scenario consumes 0.96168 Wh** [Watt hours] **which is equivalent to 13% of the capacity of an average smartphone battery** (2000 mAh, 3.7V). This value is around 3.54 times lower than other methods which use continuous GPS, because GPS lacks the property of instant availability when turning on. Wi-Fi scanning can properly detect near access points needed for SEDA 2 algorithm in the 4 second interval when it is turned on at potential stopping events determined by SEDA 1. Besides GPS's much higher consumption we also showed in Section III. B that GPS-based localization is unsuitable for the task of stopping event detection in dense urban areas due to sudden changes of its accuracy levels.

V. Conclusion

We presented a solution for stopping event detection for public transport tracking use-cases. Our method relies on continuous accelerometer readings with 5Hz frequency, from which we detect possible stopping events (SEDA 1) and 4-second periods of Wi-Fi scanning to verify if SEDA 1-determined stops are correct and if so, to localize the public traffic station found (SEDA 2).

With the idea of building Wi-Fi databases we achieved high efficiency, because we could rely solely the low-consumption accelerometer and periodic Wi-Fi scans. On a busy weekday with 12 hours of movement activity and 4 hours of public travel, SEDA algorithms consume only 13% of the capacity of a fully charged average smartphone battery, while detecting properly 82.9-89.47% of public transport stops the users traveled by.

This method not just lacks the usage of the most widely used localization technique, one of the highest consumer sensors, GPS, but solves the rarely addressed task of stopping event detection during traveling on public vehicles with the least power-hungry sensor in smartphones, the accelerometer and periodic Wi-Fi scanning in 6.66% of actual traveling time on public transport.

Acknowledgement

The publication was supported by the EITKIC_12-1-2012-0001 project, which is supported by the Hungarian Government, managed by the National Development Agency, financed by the Research and Technology Innovation Fund and was performed in cooperation with the EIT ICT Labs Budapest Associate Partner Group (www.ictlabs.elte.hu). Károly Farkas has been partially supported by the Hungarian Academy of Sciences through the Bolyai János Research Fellowship. Ármin Petkovics has been partially supported by High Speed Networks Laboratory, Budapest University of Technology and Economics.

References

[1] Kenneth J. Schmier, Paul Freda, "Public transit vehicle arrival information system," U.S. Patent No. 6 374 176 B1, Apr. 16, 2002

[2] Bulut, M. F., and Demirbas, M.: „Energy Efficient Proximity Alert on Android," Workshop on Pervasive Collaboration and Social Networking, PerCOM, 2013.

[3] Thiagarajan, A., Biagioni, J., Gerlich, T., & Eriksson, J.: „Cooperative Transit Tracking using Smart-phones," 8th ACM Conf. on Embedded Networked Sensor Systems, ACM, 2010, pp. 85-98.

[4] Wang, Y., Lin, J., Annavaram, M., Jacobson, Q. A., Hong, J., Krishnamachari, B., & Sadeh, N.: „A Framework of Energy Efficient Mobile Sensing for Automatic User State Recognition," Proc. of the 7th Int. Conf. on Mobile Systems, Applications and Services, ACM, 2009, pp. 179-192.

[5] Cardone, G., Foschini, L., Borcea, C., Bellavista, P., Corradi, A., Talasila, M., & Curtmola, R.: „Fostering ParticipAction in Smart Cities: a Geo-Social CrowdSensing Platform," IEEE Commun. Mag., 51(6), June 2013.

[6] Liu, C. H., Fan, J., Hui, P., Crowcroft, J., & Ding, G. (2013). QoI-Aware Energy-Efficient Participatory Crowdsourcing. *IEEE Sensors Jour.*, *13*(10), 3742-3753.

[7] Zhou, P., Zheng, Y., & Li, M.: „How long to wait?: Predicting bus arrival time with mobile phone based participatory sensing," in Proc. of the 10th Int. Conf. on Mobile systems, applications, and services, MobiSys'12, ACM, 2012, pp. 379-392.

[8] Yeoh, Wee-Soon, et al. "Ambulatory monitoring of human posture and walking speed using wearable accelerometer sensors." *Engineering in Medicine and Biology Society, 2008, EMBS'08, 30th Annual Int. Conf. of the IEEE*, 2008.

[9] Kwapisz, J. R., Weiss, G. M., & Moore, S. A. (2011). "Activity recognition using cell phone accelerometers", *ACM SigKDD Explorations Newslett.*, *12*(2), pp.74-82

[10] Google Support, Manage Location for your device [blog], Available: https://support.google.com/nexus/answer/3467281 [Accessed: Dec 16 2013]

[11] Kjaergaard, M. B. "Location-based services on mobile phones: minimizing power consumption." *Pervasive Computing, IEEE* 11.1, 2012, pp. 67-73.