CutiQueue: People Counting in Waiting Lines Using Bluetooth Low Energy Based Passive Presence Detection

Falk Brockmann
University of Duisburg-Essen,
Schützenbahn 70, 45127 Essen, Germany
Email: falk.brockmann@uni-due.de

Marcus Handte
University of Duisburg-Essen,
Schützenbahn 70, 45127 Essen, Germany
Email: marcus.handte@uni-due.de

Pedro José Marrón
University of Duisburg-Essen,
Schützenbahn 70, 45127 Essen, Germany
Email: pjmarron@uni-due.de

***Abstract*—Queueing systems are used to estimate the attributes of waiting lines, for example the number of people in line or the waiting time. This information helps to reduce the time spent waiting by balancing people amongst multiple lines or aids in the decision making about opening a new line. However, current queueing systems are often mounted at fixed positions, require user participation, or need time consuming manual calibration after each layout change. In this work, we introduce CutiQueue, a flexible and portable queueing system that is battery powered, needs no manual calibration or user participation. CutiQueue is based on passive presence detection using the RSSI of low cost Bluetooth Low Energy transceivers. We analyze the performance of the system in experiments using a prototype implementation to estimate the number of people waiting as well as the length of the line. The experimental evaluation indicates an average accuracy of 97.96% when compared to counting people walking or standing in the waiting line using a laser barrier.**

I. Introduction

Waiting lines are part of daily life. Examples include coffee shops, supermarkets, airports, or entertainment facilities such as amusement parks or music festivals. A waiting line has a service point at which the desired resources, e.g. plane tickets or a cup of coffee, are distributed, and a waiting area for people to spend their time until the staff at the service point can serve the next customer. In facilities dealing with hundreds of people every day, waiting areas are typically indicated by a set of portable barrier poles that are connected by retractable belts. This way, the layout of the waiting line can easily be changed by moving the barrier poles to different positions. For example, at an airport, adjusting the layout can be necessary to guide passengers to a different check-in counter, or at an amusement park, to a newly opened ticket booth. Knowing the performance attributes of the waiting line, like the number of people in the line, can help making decisions about when to open secondary service points to reduce the overall waiting time.

However, current queueing systems to track the number of people in the line are often installed at fixed positions and cannot be moved at all or need time-consuming manual recalibration after each layout change. This reduces the queueing systems capabilities to record performance attributes and thus negatively affects the waiting time.

The goal of this work is the creation of a portable queueing system, capable of counting the people in line, that is easy to deploy, has a high detection accuracy and has no need for manual calibration. Recent work in the field of RSSI-based passive detection has shown that battery powered radio nodes can be used to accurately detect human presence in the radio link between the nodes [1], [2].

In order to achieve this goal, we equip wooden barrier poles with affordable radio transceivers and create a prototype queueing system called *C*ommunication *Uti*lity *Queue* (*CutiQueue*). As communication technology we use Bluetooth Low Energy (BLE) because of its stable Received Signal Strength Indicator (RSSI), to achieve a high detection accuracy as shown in our previous work [3].

The contributions of the paper are the development and implementation of the necessary communication infrastructure between the nodes of the queueing system based on the BLE protocol and the creation of the CutiQueue algorithm that counts the persons passing and waiting in the line. Furthermore, we adapt our previous work on presence detection with BLE [3] for this application scenario and evaluate the queueing system in a set of experiments.

This paper is structured as follows: Section II provides a brief overview of the related work in the fields of queueing systems and RSSI-based passive presence detection. In Section III we present the BLE based detection algorithm, the communication infrastructure, and the CutiQueue algorithm. We evaluate our approach in Section IV. Section V concludes the paper.

II. Related Work

This section gives an overview on existing work in the areas of queueing systems and RSSI-based detection algorithms for human presence.

Queueing Systems are used to analyze the properties of waiting lines. Current queueing systems can be classified into participatory systems, in which the user is actively providing information to the system, and autonomous systems, which gather the information independently. Examples for participatory systems are [4] and [5], in which users carry hand-held

devices, i.e. smartphones, to interact with the queueing system. The information needed to estimate the waiting lies properties can either be extracted from the smartphones wireless capabilities [4], [6], [7], or from one of the smartphone sensors, e.g. the accelerometer [5].

In [4] a special, traffic generating app is installed on the devices. Messages send from the app are received and evaluated by a single signal monitor at the service point of the waiting line. Depending on the strength of the received WIFI signals, the system estimates whether a person is waiting in line, approaching the service point or leaving the line. The higher the signal strength the closer the person is assumed to be to the service point. The number of connections can be used as an indicator of the number of people in the line.

As alternative to use the RSSI from the WIFI signals the accelerometer data of a smartphone is used in [5]. The data of devices in the waiting line is collected by an app and transmitted to a central server for analysis. There it is classified into tranquil periods of waiting and periods of moving forward in the line, i.e. "shuffling forward".

Both techniques utilize smartphones to transmit the gathered data to the queueing system. However, while smartphones are largely common in urban areas, with coverage values nearing 100% [6], it still is difficult to provide the necessary incentive for users to participate [8] in such a system. The lack of interest, issues regarding privacy or simply the missing knowledge about the existence of the system can prevent users from participating, negatively influencing the accuracy.

Autonomous queueing systems do not require users to carry devices but instead utilize previously installed infrastructure. Common types of those systems are camera or beam-type systems, e.g. laser-beam, light-beam or infra-red. Camera based systems use computer vision-based algorithms on the visual input of the cameras to estimate the number of persons in an area. However, cameras are often installed at fixed positions with a limited field of view and are not flexible to changes in the layout of a waiting line. Additionally, constant video monitoring raises privacy issues, since every user can be identified.

Beam-type systems detect people entering or leaving the system by counting the interruptions of a laser- or light-beam received by a light sensor. When a user interrupts the beam the change in the luminance is registered and the estimated number of persons in the waiting line is increased. However, beam-type systems relay on the accurate calibration of the laser-beam aiming at the light sensor. A layout change in a beam-type system requires time-consuming calibration every time the system is altered. This reduces the flexibility of the system. Another issue beam-type systems share is their sensitivity to sunlight. Sunlight directly hitting the light sensor can increase the received luminance to a level on which passing persons are not recognized anymore, causing the system to miss detection events.

RSSI-based queueing systems on the other hand do not share these issues. They are not influenced by visible light, people cannot be identified, and the radio waves are transmitted omnidirectional, which removes the need for calibration after a layout change. For example, in [9] an autonomous queueing system used to estimate road occupancy and traffic queue lengths is introduced.

By studying the fluctuations of the RSSI of a radio link, it is also possible to passively detect the presence of a person. Algorithms for passive presence detection can coarsely be classified into mean RSSI-based and RSSI variance-based techniques [10]. An early work using the mean RSSI-based technique is [11], in which the influence of the human body on a radio link is analyzed. If the currently measured RSSI is one standard deviation below the mean RSSI, a detection event on the radio link will be assumed. Building on this in [2] and [12] a grid of sensor nodes is used to detect a person moving in a room. The concept, introduced in [12], of Radio Tomographic Imaging (RTI) is then used to track and visualize the movement of the person. In [1] and [13] these results are extended by using a variance-based algorithm to cope with fluctuations in the RSSI caused by long term changes in the environment.

The challenges of passive presence detection on BLE radio links are addressed in our previous work by extending a mean RSSI-based and a RSSI variance-based algorithm to cope with different BLE channels [3]. In this paper we further explore this by creating a hybrid algorithm using both techniques simultaneously. We then use the results of the hybrid algorithm as input for the CutiQueue algorithm to count people in a waiting line scenario.

Thus, the approach in this work uses the advantages of flexible, omnidirectional passive RSSI-based detection to create an autonomous queueing system that does not need user participation, is easy to deploy and is not limited to fixed sensor positions.

III. Approach

This section describes the techniques used to create the CutiQueue prototype queueing system. It is split in three parts. The first part addresses detecting people using the RSSI-based hybrid passive presence detection algorithm and the advantages and challenges in using BLE. The second part focuses on the communication infrastructure for the transmission of the detection results and implementation details of the prototype queueing system, while the third part explains the CutiQueue algorithm for counting people in the waiting line.

A. BLE Based Detection

In order to create the prototype queueing system we establish multiple radio links between radio transceivers on opposite sides of the waiting area. Messages are constantly being sent between the transceivers generating RSSI samples to be analyzed by the detection algorithm. The space between the transceivers is hereby called the monitored area. Traditionally, protocols based on the 802.15.4 standard or WLAN are used to establish these links [11]. However, along with the increasing popularity of smartphones in recent years new standards for wireless communication, have been introduced, like BLE [14].

TABLE I
STATES OF THE RSSI-BASED DETECTION ALGORITHM

State	Description
0	No detection - empty link
1	Detection - person in link
2	No detection with reduced credibility
3	Detection with reduced credibility
4	Error
5	Calibration

The BLE protocol is designed with the goal of supporting short range data transmission at low energy cost. This design is especially aimed at small, battery powered devices such as radio beacons which are periodically broadcasting messages. The network connection is established between a central device, like a smartphone, and a peripheral, like a wearable or a beacon.

BLE uses 40 channels separated by 2MHz in the spectrum between 2402 MHz and 2480 MHz. Three of the channels are advertising channel used to broadcast advertisement messages for establishing a BLE connection, the others are used for data transmission. In addition to their intended purpose the advertising channels can also be used to create radio links supporting passive presence detection. The advantages of BLE compared to WIFI include a very stable RSSI on the radio links, which can be used to improve the detection accuracy for presence detection algorithms [3].

The two algorithms introduced in our previous work [3] are used as foundation for the prototype queueing system and work with RSSI samples of a single radio link. Both algorithms have been tested with transceivers at different positions and distances between 0.5m and 4m to ensure high detection accuracy in a flexible set-up. The first algorithm is using RSSI variance-based detection. The RSSI variance is computed from a sliding window containing the newest RSSI samples. If the RSSI variance is above a calibrated threshold, a detection event will be triggered. Since variance in the RSSI can be caused by the movement of a person, the variance-based algorithm is well suited to detect a walking person in the monitored area with an average accuracy of 99%. However, the algorithm fails to robustly detect people standing motionless, not causing any variance in the RSSI. The second algorithm is using mean RSSI-based detection. The mean RSSI is computed as a weighted sum of all previous RSSI samples. If the mean RSSI is below a threshold based on the attenuation of a radio link caused by a person, a detection event will be triggered as well, with an detection accuracy of at least 92%. Its benefits lie in robustly detecting standing people and in a faster response time compared to the variance-based algorithm. The output of each algorithm is the current detection state on a link. An overview of the different states can be found in Table I.

The states differentiate between "*detection*" or "*no detection*" and the credibility of the estimation. The credibility is based on the quality of the link. If the quality is below a weak link threshold (WLT) introduced in [3], the credibility of the detection will be reduced.

1: $detectionState = meanAlgorithmResult$
2: **if** $(detectionState = 0) \vee (detectionState = 2)$ **then**
3: **if** $(RSSI < WLT) \wedge (varAlgorithmResult = 1) \wedge (previousVarAlgorithmResult = 0)$ **then**
4: $detectionState = 3$
5: **end if**
6: **end if**
7: **return** $detectionState$

Fig. 1. RSSI-based hybrid detection algorithm.

In this work, the mean RSSI-based detection algorithm serves as foundation for the hybrid detection algorithm. This algorithm achieves good results when analyzing the attenuation of a radio link, but neglects the influences of variation in the RSSI. To aid the mean RSSI-based algorithm in those cases, the RSSI variance-based algorithm is used. Since the variance of a radio link is influenced by human movement, this factor is added to the event detection. The decision mechanism behind the hybrid algorithm can be seen in Figure 1.

If the previous result of the mean RSSI-based algorithm is state 0 or state 2 (*no detection*), but the observed link is below the WLT, a detection event with reduced credibility can be triggered. However, this will only be the case if the variance-based algorithms result is state 1 (*detection*) and it is a newly detected event, which means, the previous result of the variance-based algorithm was state 0. This way the accuracy of the purely mean RSSI-based algorithm is increased, while the benefit of accurately detecting standing people is maintained.

B. Data Transfer

In BLE data transmission is handled by the Generic Access Profile (GAP). GAP is establishing connections using advertisement messages followed by a handshake procedure. Afterwards, data messages can be send on the data channels. However, RSSI-based passive presence detection can be implemented using the RSSI of any periodically send message, e.g. the advertising message, eliminating the need of an actual BLE connection. Keeping this in mind, the CutiQueue queueing system consists of three different applications installed on BLE transceivers and a fourth application installed on a PC. The transceivers are either configured to act as beacon, as node running the detection algorithm, or as data sink. To further reduce the data transmission time for the connection between node and sink a connectionless approach is used here as well. An example of the queueing systems structure in a deployment with four links can be seen in Figure 2.

1) Beacon: The beacon application periodically broadcasts *beacon messages* every 100ms. Each message is broadcast on one of the three advertising channels. It is received and its RSSI is analyzed by the node application. However, while the RSSI on each BLE advertising channel is very stable, the RSSI level between the three channels can largely be different [15]. To avoid mixing the RSSI from different channels and thus decreasing the detection accuracy, the detection algorithms

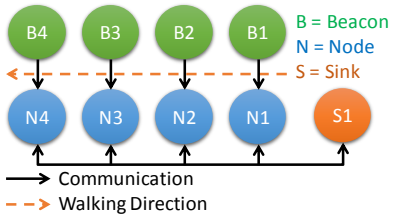

Fig. 2. Deployment of the CutiQueue prototype.

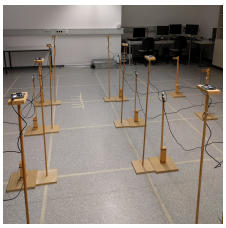

Fig. 3. Lab Set-up.

Beacon Message

5Byte Header	1Byte Length	4Byte Debug Counter	1Byte Channel Information

Event Notification Message

5Byte Header	1Byte Length	1Byte Number of Events	12Byte Event Data (3x4 Byte: State,BeaconID,RSSI,Difference)	1Byte Unique ID of Last Event

Acknowledgement Message

5Byte Header	1Byte Length	1Byte ACK (Node ID)	1Byte Unique ID of Last Event

Fig. 4. Structure for the beacon, event notification and acknowledgment messages.

have to handle the advertising channels independently. Yet, the BLE radio stack often is a closed system and has no configuration possibilities [16]. Thus, the information about the current advertising channel is hidden from the application programmer. To solve this issue the beacon application includes the information about the advertising channel in the payload of every message. This is done by stopping the, otherwise constantly running, advertisement mode after each message, setting the advertising channel to a specific one, and sending the next message including the updated information. The GAP message structure is hereby retained such that the messages can still be processed by arbitrary BLE devices. The message structure can be seen in Figure 4.

2) Node: The node application has two tasks. First, it continuously scans for new beacon messages, secondly, it analyzes new RSSI samples using the hybrid detection algorithm and forwards any changes in the link state to the sink. For this, the node is configured to run the BLE scanning mode after its initialization. While in this mode, it will receive all messages broadcast on the advertising channels and filter them by the included node ID. When a message has been received containing its own node ID, the node differentiates between beacon messages or *acknowledgment messages* send by the sink. Is the received message a beacon message the node checks, whether the detection algorithm is still in the automatic calibration phase, introduced in [3]. The calibration phase ends when at least one message was received on each advertising channel and is used to ensure that the initial values of the detection algorithm are collected in a tranquil phase without human presence in the link. The duration of this phase depends on the scan interval of BLE after which the channels are switched and lies between 10.25s to 20.49s in our experiments. During the calibration phase the detection algorithm gathers data, RSSI variance and mean are computed, but no detection state information is generated. When the detection algorithm is not in the calibration phase, it analyzes the RSSI of the received beacon message and computes the current detection state. Only if there is a change in the detection state of the link, a new *event notification message* will be generated and send to the sink. This is handled by sending one advertisement message on each advertising channel and starting an *acknowledgment timer*. The event notification message can contain one event with up to two older events piggybacking the same message. Each event contains its detection state, beacon and node ID as well as the time difference to the previous event if piggybacking. Additionally, the event notification message includes the number of contained events and the unique ID of the last event waiting to be acknowledged. The message structure can be seen in Figure 4. After the event notification message was sent, the BLE scan mode for new messages is started again.

The acknowledgment timer has a timeout of 1s. If no acknowledgment message is received during this time a new event notification message will be sent every 1s while the BLE scan is running. The event notification message can be the same message as before or an updated version with the old event piggybacking, if a detection state change occurred. If an acknowledgment message has been received while the acknowledgment timer is running, it will be stopped, and the acknowledgment is analyzed. If all events are acknowledged by the unique ID of the last event included in the received acknowledgment message, they will be deleted from the event notification message. If no event is acknowledged, the currently running BLE scan will be stopped and the advertising of the event notification message will be resumed. After the acknowledgment has been handled the node application returns to its initial BLE scanning mode.

3) Sink: The sink applications main purpose is to forward data to the PC application and to acknowledge every received event notification message with an acknowledgment message. For this, the sink application is also set into the scanning mode of BLE after its initialization. When an event notification message is received, the scan is stopped and the message is analyzed. The sink filters the events included in the message and only forwards those, that have not been forwarded before, to the PC application. Afterwards, the advertisement mode is started, broadcasting a single acknowledgment message on each channel. The acknowledgment message includes the node ID of the node that sent the event notification message and the unique ID of the last event received from that node. This way, only the node matching node ID will consider the acknowledgment and set all of its events up to the transmitted unique ID as acknowledged. Then, after broadcasting the acknowledgment message, the BLE scan on the sink is started again.

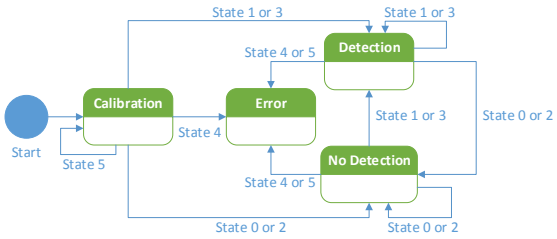

Fig. 5. Finite-state machine of the CutiQueue algorithm for one link.

C. CutiQueue Algorithm

The CutiQueue algorithm is used to estimate the number of people currently waiting in line and runs as an application on a PC. The aim is to design an algorithm, that can use the information from all links to create a synergy between the separate event detections. If a person crossing a link has been missed and no detection event was created, the information from other the links should compensate this, if possible. The CutiQueue algorithm is based on the information from the underlying hybrid detection algorithm and must be able to cope with possibly faulty information.

1) Finite-State Machine: The CutiQueue algorithm is executed whenever a new message arrives from the sink. Each message is sorted and matched to the link it was received on. Then, the detection state of the link is updated following the finite-state machine in Figure 5. The state included in the current message is compared with the previous state of the same link, which is initialized with the calibration state 5. The new state is handled depending on the previous state.

- If there is no change in the state, the message will be ignored by the state machine.
- If the credibility of the detection state changes, either from 0 to 2, 1 to 3, or vice versa, the link state will be updated.
- If the detection state changes from *no detection* to *detection* the detection state will be updated, and two timers will be started. The first timer is the *blockedLinkTimer*, which is used to keep track of a detection events duration on a link. If a link is occupied for a time longer than set by the blockedLinkTimer, the link will be considered "blocked". If all links from the end of the waiting line are blocked, it will be seen as an indicator that people are waiting and the line is filled up to this link. The second timer is the *recoveryTimer*. Since the system is working on possibly faulty input, it can drift away from the accurate number of people counted during its runtime with no way of recovery. To avoid this, the recoveryTimer is used to set the number of persons currently in the system to zero in phases on tranquility, i.e. when no person is detected in the waiting area. The timer will be triggered, if there is no detection event on all links in the complete system for a set period of time. Lastly, a third timer, the *freeLinkTimer*, will be stopped, if the current detection event is recorded on the last link at the end of the waiting line. The freeLinkTimer timer is used to estimate how many people have left the queue. It will be triggered, if the last link of the system, so the first one to be blocked when people are waiting, records no detection event for a set period of time. It is stopped, whenever a detection event occurs on the last link.
- If the detection state changes from *detection* to *no detection* again, the detection state will be updated. The blockedLinkTimer will be stopped and the link will be set to "not blocked". Furthermore, the freeLinkTimer will be set, if the observed link is the last link of the waiting line.

The duration of the blockedLinkTimer, recoveryTimer and freeLinkTimer are discussed in Section IV.

After the state of the current link has been updated, the influence of any new detection event (state 1 or 3) on the estimated number of people in the waiting line is checked. However, due to the non-uniform influence on the RSSI caused by human arm and leg movement when crossing a radio link, the same person can trigger multiple detection events. This can occur especially when entering or leaving a link. To cope with this, an *mergeInterval* is introduced possibly grouping multiple detection events into one. The mergeInterval is based on the end of the last detection event on a link and is updated every time a new event follows state 1 or 3. If the new event is in the merge interval of the previous event on the same link, they will be combined into one event with increased duration and the new event will be discarded. If the new event is not in the mergeInterval of the previous event, it will be checked whether a new person is to be estimated in the waiting line.

2) People Counting: The queueing system stores a list of *estimations* to count the current number of people in line. Each estimation has a slot for each link in the queueing system. If a new detection event is found, the slots for the respective link will be checked by iterating through the list of all estimations. If one or more estimation with an empty slot for the link exist, the detection event will be set into the empty slot of the first estimation found. If all slots for the respective link are occupied, a new estimation will be created. In both cases the detection event is discarded afterwards.

A global timer is used to update the estimate of people in the waiting line, that is periodically triggered after a set duration. This *countingTimer* iterates through the complete list of estimations every time it is triggered. For each slot containing a detection event, points are awarded to the estimation depending on the state of the detection event. For a state 1 detection event 50 points are awarded, for a state 3 detection event 34 points. Afterwards the points are added up to a total. If the total number of points is higher than a threshold of 100 points, an additional person will be counted by the queueing system. This way at least two links with strong signal quality or three links with weak signal quality are needed for a new person to be counted, reducing false positives.

Additionally, the threshold is influenced by the number of blocked links. If links are blocked, they are no longer available

to generate new detection events. Since the number of blocked links reduces the potentially available number of links to count people, the threshold of 100 points is reduced by an amount of $(1/numberOfLinks) * 100$ points for each blocked link.

If a new person is counted, the respective estimation will be marked as such, but not yet removed from the list. This is done to prevent the same person being counted multiple times. Only after a estimation has an entry in each slot and has been marked as counted, it is removed. As a last step, the number of persons identified as leaving by the freeLinkTimer is subtracted from the computed total. The subtrahend is displayed as the result of the CutiQueue algorithm and is the estimate of the number of people currently in the waiting line.

The values for the countingTimer and mergeInterval are discussed in Section IV.

IV. Evaluation

To evaluate the CutiQueue queueing system, we perform multiple experiments under lab conditions to test the systems robustness and accuracy, as seen in Figure 3. For this, we build a prototype queueing system by mounting 8 BLE transceivers on wooden poles. We use the ATMEL Xplained Pro evaluation platform as transceivers featuring a ATMEL SAMB11 Bluetooth 4.1 module. In the prototype system the transceivers are powered via USB cable which are also used for data collection. The barrier poles are ordered in a grid of 1x4 meters indicating the waiting area in a corridor like fashion, following the deployment seen in Figure 2. All beacons are placed at one side of the corridor, all nodes on the other side. For our evaluation, radio links are only established between the beacon and the node on exactly opposite sides of the corridor. Doing so, our prototype system observes four radio links crossing the waiting area orthogonally to simulate part of a waiting line in e.g. an airport scenario. Additionally, a transceiver is used as sink and connected to a laptop running the PC application with the CutiQueue algorithm.

We compare the results of the hybrid detection algorithm and of the CutiQueue algorithm with a ground truth generated from a laser barrier. In order to set-up the laser barrier, we use the light sensors of smartphones and aim wall-socket powered laser-pointers at them, one pair per link. While it generates an accurate ground truth, the laser barrier has a high set-up time of up to 35s per link and is sensitive to the influence of ceiling lamps or sunlight. The laser-pointers are positioned in a way, that when a radio link is crossed, the respective laser-beam is interrupted as well. Both, the hybrid detection algorithm and the laser based ground truth system then generate a detection event which is forwarded to the same laptop running the PC application. There, each event is combined with a timestamp, such that the events can be set in direct relation to each other. Due to the nature of the RSSI-based detection, it takes a longer time to detect a person, than using the laser based ground truth. When analyzing the time difference between a detection event and the ground truth event, we find that 66% of all received detection events are recorded within 1s after the respective ground truth event. 98% of all detection events are recorded within 2.5s after the ground truth event. For our analysis we double this time and define an detection event as match, if it was recorded within an interval of 5s to the ground truth.

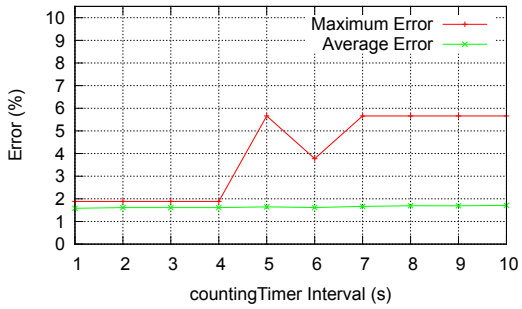

Fig. 6. Error of the CutiQueue algorithm with different interval for the countingTimer.

We use the precision, recall and F1-Score of the event detection as metrics to evaluate the accuracy of the hybrid detection algorithm. The precision measures how many of the detected events are actual matches. It is computed using Equation 1.

$$precision = \frac{\#truepositives}{\#truepositives + \#falsepositives} \quad (1)$$

The recall measures how many matches have been missed. It is computed using Equation 2.

$$recall = \frac{\#truepositives}{\#truepositives + \#falsenegatives} \quad (2)$$

The F1-Score combines both an is computed using Equation 3.

$$\textit{F1-score} = 2 * \frac{precision * recall}{precision + recall} \quad (3)$$

The error between the current estimated number of people in the waiting line and the actual number of people obtained from the ground truth is computed as metric to evaluate the CutiQueue algorithm. This is done by comparing the two values every time a new message arrives and computing the error in percent using Equation 4.

$$\%_{error} = |\frac{\#estimated - \#actual}{\#actual}| * 100 \quad (4)$$

A. Parameter Configuration

In order to configure the parameters for the CutiQueue algorithm, we first perform an experiment *EX1* with the hybrid detection algorithm. During one run of the experiment, the four radio links of the prototype queueing system are crossed by a person in consecutive order. Detection events and ground truth events are recorded and forwarded to the PC application. The experiment run is repeated 53 times. The results of the hybrid detection algorithm and its comparison to the ground truth can be seen in Figure 7. For clarity reasons, both state 1 and state

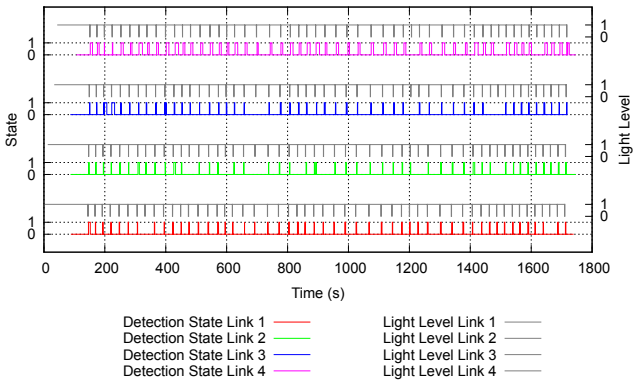

Fig. 7. Detection events compared to ground truth for EX1.

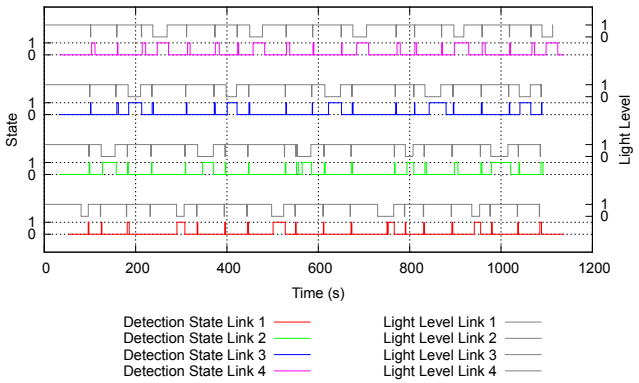

Fig. 8. Detection events compared to ground truth for EX3.

TABLE II
PERFORMANCE PARAMETERS FOR THE RADIO LINKS IN EX1.

Link	Precision (%)	Recall (%)	F1-Score (%)
1	100	96.2	98.1
2	97.9	86.8	92.0
3	100	92.5	96.1
4	100	94.3	97.1

3 are plotted as '1', and state 0 and state 2 as '0'. The light level of the ground truth is set to '0' for detection, and '1' for no detection. The detection algorithm achieves an accuracy of 98.1%, 92% 96.1% and 97.1% on the respective links from start to end of the waiting line, thus reaching a detection accuracy of 95.8% averaged over all links. The results can be seen in TableII. Afterwards, the CutiQueue algorithm is run on the detection results to estimate the best values for the parameters.

1) Merge Interval: First, the merge interval for the CutiQueue algorithm is configured. For this, we take a look at the duration of the detection events. While the average duration of a ground truth event is 0,622s, the average duration of a detection event is 2,956s, since the radio link is influenced by human presence for a longer time. Analyzing the results of EX1 we observe that by setting the merge interval to a value of 5s actual detection events caused by a person crossing a link are no longer wrongly identified as false positives. Thus, the merge interval is set to a duration of 5s for our further experiments.

2) countingTimer Interval: Testing the influence of the countingTimer by using different values from 1s to 10s, we observe that the countingTimer interval has no influence on the number of counted people at the end of the experiment (as long as it is executed at least once). However, the maximum and average error increase from 1.9% at 1s to 5.6% at 10s and 1.58% at 1s to 1.71% at 10s respectively, as seen in Figure 6. This is caused by the longer duration between two executions of the people counting process. Since the duration is longer there is more time for people to enter the system without being counted, increasing the time in which the number of counted people is inaccurate, despite them being counted later on. Because of this, the countingTimer interval is set to 1s.

3) blockedLinkTimer Interval: When determining the merge interval we assume that events on a link can belong to the same detection event for a duration of up to 5s, when the link is crossed. In order to clearly differentiate between crossing and standing in the link we double this time and assume that a link is blocked after a detection event that lasts at least 10s. For this reason the blockedLink interval is set to 10s.

4) freeLinkTimer Interval: Following the same logic as before, if there are no new events in a duration of 5s we assume that the link is free and currently no person is standing in it. Thus, the freeLinkTimer interval is set to 5s.

5) recoveryTimer Interval: The duration of the recoveryTimer interval needs to be in the balance point between two extremes. If the interval is set to high, error in the estimation will accumulate normally, as if the recovery mechanism would not be existing. If the interval is set to low, the number of counted people will always be reset to zero after a person has been counted, despite there possibly being multiple people in the line. In order to avoid both, the recoveryTimer is set to only fire, if there is an interval of 10s since the last detection event in which the complete waiting line is undisturbed and no new detection events are recorded on any link.

B. Queueing System Evaluation

We perform additional experiments to analyze the accuracy of the prototype queueing system during its operation. For this, we first repeat the experiment performed to calibrate the parameters. A person is walking through the waiting line 62 times, crossing the links consecutively. To normalize the error we divide it by the number of people counted by the ground truth after the end of the experiment.

Figure 9 shows the error experienced in EX2 without the use of the recoveryTimer. As seen in the figure, the error slowly accumulates over time, reaching a final value of 4.839% at the end of the experiment. Figure 10 shows the error of EX2 while the recoveryTimer is used. Since the system can recover itself in tranquil phases, when no person is in the waiting line, the experiment is concluded with no error and an average error of

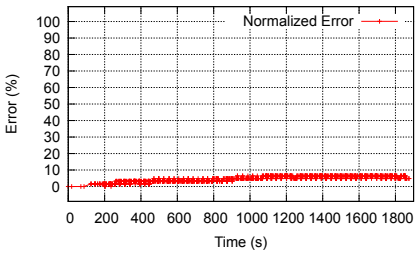

Fig. 9. Error of EX2 without recoveryTimer.

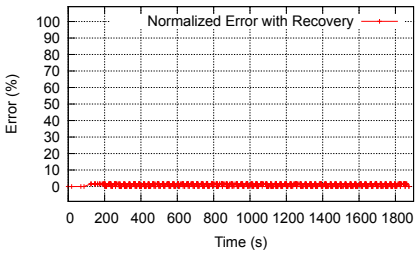

Fig. 10. Error of EX2 with active recoveryTimer.

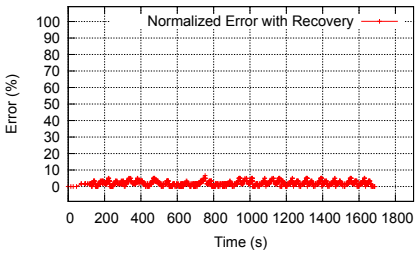

Fig. 11. Normalized error of EX4.

1.093% over time with a standard deviation of 0.754%. This equals an accuracy of 98.91%. Out of the 62 times the person is walking through the prototype queueing system, 59 times have been counted at the end of EX2.

In the third experiment *EX3* the detection of blocked links is tested. A person walks through the queueing system 20 times, each time stopping at a different link and blocking it. The results of the hybrid detection algorithm of EX3 can be seen in Figure 8. 19 out of the 20 times a link is blocked are detected and 20 of 20 persons are counted in EX3.

Next, we test the prototype queueing systems capabilities, for situations when multiple persons are in the waiting line. For this, we perform an experiment *EX4*, in which three persons enter the waiting line after another. The first person entering walks through the complete line and waits at its end. Afterwards, the second persons enters and waits behind the first person, in the monitored area of the next link. Finally, the third person enters, and waits in the link after the second person. When all have entered, the first person leaves the system. The second and third person each advance to the next link and wait. Next, the second person leaves the system, while the third person advances to wait in the last link. At the end, the third person leaves the link.

The experiment is repeated 20 times, so that the waiting line is passed through 60 times in total. The results of EX4 can be seen in Figure 11. During the experiment, 109 out of the 120 times a link was blocked are detected, and 54 of the 60 persons walking through the system are counted. The CutiQueue queueing system achieves an average error of 2.042% over time with a standard deviation of 1.309%, which equals an accuracy of 97.96%.

This demonstrates that the combination of the hybrid detection algorithm with fast, BLE-based data transmission and the CutiQueue algorithm can be used under realistic conditions to count the people in a waiting line.

V. Conclusion

In this work, we developed a prototype queueing system to count the number of people in a waiting line. The system is based on RSSI-based passive detection and no user participation is required. It is easy to deploy, has high portability and needs no manual calibration. The system achieves an accuracy of 98.91% when analyzing a single person passing through it, and an accuracy of 97.96% when analyzing a waiting line in which there are multiple people simultaneously.

References

[1] J. Wilson and N. Patwari, "See-through walls: Motion tracking using variance-based radio tomography networks," *Mobile Computing, IEEE Transactions on*, vol. 10, no. 5, pp. 612–621, 2011.

[2] O. Kaltiokallio and M. Bocca, "Real-Time Intrusion Detection and Tracking in Indoor Environment through Distributed RSSI Processing," *2011 IEEE 17th International Conference on Embedded and Real-Time Computing Systems and Applications*, pp. 61–70, aug 2011.

[3] F. Brockmann, R. Figura, M. Handte, and P. J. Marrón, "RSSI based passive detection of persons for waiting lines using Bluetooth Low Energy," in press.

[4] Y. Wang, J. Yang, Y. Chen, H. Liu, M. Gruteser, and R. P. Martin, "Tracking human queues using single-point signal monitoring," *Proceedings of the 12th annual international conference on Mobile systems, applications, and services - MobiSys '14*, pp. 42–54, 2014.

[5] T. Okoshi, Y. Lu, C. Vig, Y. Lee, R. K. Balan, and A. Misra, "Queue-Vadis: Queuing Analytics using Smartphones," *Proceedings of the 14th International Conference on Information Processing in Sensor Networks - IPSN '15*, pp. 214–225, 2015.

[6] Y. Malinovskiy, N. Saunier, and Y. Wang, "Analysis of pedestrian travel with static bluetooth sensors," *Transportation Research Record: Journal of the Transportation Research Board*, no. 2299, pp. 137–149, 2012.

[7] M. Versichele, T. Neutens, M. Delafontaine, and N. Van de Weghe, "The use of bluetooth for analysing spatiotemporal dynamics of human movement at mass events: A case study of the ghent festivities," *Applied Geography*, vol. 32, no. 2, pp. 208–220, 2012.

[8] S. Sen, D. Kim, S. Laroche, K.-H. Kim, and J. Lee, "Bringing CUPID Indoor Positioning System to Practice," in *Proceedings of the 24th International Conference on World Wide Web - WWW '15*. New York, New York, USA: ACM Press, 2015, pp. 938–948.

[9] R. Sen, A. Maurya, B. Raman, R. Mehta, R. Kalyanaraman, N. Vankadhara, S. Roy, and P. Sharma, "Kyun queue: a sensor network system to monitor road traffic queues," *Proceedings of the 10th ACM Conference on Embedded Network Sensor Systems (SenSys)*, pp. 127–140, 2012.

[10] F. Brockmann, S. Jungen, C. Y. Shih, M. Handte, and P. J. Marrón, "Accurate Event Detection and Velocity Estimation in Wireless Environments," in *Computer Science and Information Systems (FedCSIS), 2016 Federated Conference on*, vol. 8, oct 2016, pp. 1057–1066.

[11] S. Hussain, R. Peters, and D. Silver, "Using received signal strength variation for surveillance in residential areas," *Data Mining, Intrusion Detection, Information Assurance, and Data Networks Security 2008*, 2008.

[12] J. Wilson and N. Patwari, "Radio tomographic imaging with wireless networks," *Mobile Computing, IEEE Transactions on*, vol. 9, no. 5, pp. 621–632, 2010.

[13] O. Kaltiokallio, M. Bocca, and N. Patwari, "Follow @grandma: Long-term device-free localization for residential monitoring," *Proceedings - Conference on Local Computer Networks, LCN*, pp. 991–998, 2012.

[14] Bluetooth Special Interest Group (SIG), "Core Version 5.0," 12 2016. [Online]. Available: https://www.bluetooth.com/specifications/bluetooth-core-specification

[15] Y. Zhuang, J. Yang, Y. Li, L. Qi, and N. El-Sheimy, "Smartphone-based indoor localization with bluetooth low energy beacons," *Sensors (Switzerland)*, vol. 16, no. 5, pp. 1–20, 2016.

[16] M. Spörk, C. A. Boano, M. Zimmerling, and K. Römer, "Bleach: Exploiting the full potential of ipv6 over ble in constrained embedded iot devices," 2017.