Car-to-Pedestrian Communication with MEC-Support for Adaptive Safety of Vulnerable Road Users

Quang-Huy Nguyen[a], Michel Morold[b], Klaus David[b], Falko Dressler[*,a]

[a] *Dept. of Computer Science and Heinz Nixdorf Institute, Paderborn University, Germany*
[b] *Faculty of Electrical Engineering / Computer Science, University of Kassel, Germany*

Abstract

Car-to-Pedestrian (Car2P) communication has a huge potential for preventing potential accidents between pedestrians and vehicles by exchanging context information. Similar communication principles have been explored for Car-to-Car (C2C) communication since quite a while and are now being deployed as an additional safety mechanism. Unfortunately, Vulnerable Road Users (VRUs) such as pedestrians and bicyclists are not yet covered by these systems. Looking at pedestrians, we explore possibilities for the exchange of relevant safety mechanisms between pedestrians and cars making use of readily available communication capabilities of current LTE networks. Necessary activity and collision detection algorithms have to be run on the users' smartphones in order to determine the criticality of the situation. In order to improve both the overall system latency and the energy efficiency of the smartphone applications, we suggest the use of Multi-access Edge Computing (MEC). We implemented the system to perform measurements on the smartphone side as well as extensive simulations on the networking side. Our results clearly show the advantages of our concept and the integrated MEC approach.

Key words: Vulnerable road users, multi-access edge computing, offloading, car-to-pedestrian communication

1. Introduction

In recent years, cooperative collision detection has emerged as a complementary approach to traditional safety solutions using vehicle sensors like cameras, laser scanner, and RADAR to detect nearby objects. Different from the sensor-based approach, which requires a direct Line-Of-Sight (LOS) between vehicles and detected objects, cooperative collision detection can even operate in Non-Line-Of-Sight (NLOS) scenarios by exchanging movement information between entities via wireless communication. In the context of safety systems for Vulnerable Road Users (VRUs) like pedestrians and bicyclists, vehicles exchange information with VRUs' mobile smart devices, e.g., smartphones, tablets, or wearable devices [1, 2, 3, 4].

The general concept of a VRU safety system is shown in Figure 1 (extending our earlier work in [5]), in which vehicles and VRUs have two possible options for data transmission, namely direct Device-to-Device (D2D) communication and an infrastructure-based approach. As an initial idea, D2D technologies, such as Wi-Fi Peer-to-Peer (P2P) or Dedicated Short Range Communication (DSRC), are designed to serve the need of cooperative safety systems, which allows messages to be directly transferred from sender to receiver. In particular, DSRC has been assumed for Car-to-Car (C2C)/Vehicle-to-Vehicle (V2V) and Car-to-Everything (C2X)/Vehicle-to-Everything (V2X) communication and is likely to be available for smartphones in the next few years [6, 7]. However, these D2D technologies do not offer good coverage and range, i.e., less than 200 m for Wi-Fi Direct and less than 1000 m for DSRC, especially around intersections in urban areas [8, 9]. To overcome these limitations, cellular technology is proposed as an alternative.

Cellular V2X (C-V2X), or more specific LTE-V2X, includes two interfaces: the LTE interface (named Uu), that supports the communication between vehicles/end-devices and mobile base stations and the newer D2D interface (named PC5), that enables V2V communications based on direct LTE sidelink [10, 11]. C-V2X is indicated to have many advantages in different aspects, such as communication range, performance, and reliability, especially with the current evolution from LTE to 5G [12]. PC5, like DSRC, has been studied by several existing works and is deployed mainly for vehicles [13, 10, 11]. This direct V2X communication can significantly reduce the latency and is expected to become superior to DSRC [12]. Meanwhile, the Uu radio interface, which is already available for smartphones, still remains an important part of the next generation wireless V2X system with the ability to support long-range communication. However, the feasibility of applying this interface and network infrastructure to provide an additional approach for different V2X use cases

*Corresponding author.
Email addresses: nguyen@ccs-labs.org (Quang-Huy Nguyen), michel.morold@comtec.eecs.uni-kassel.de (Michel Morold), klaus.david@comtec.eecs.uni-kassel.de (Klaus David), dressler@ccs-labs.org (Falko Dressler)

Article published in Elsevier Computer Communications 150 (2020) 83-93. (10.1016/j.comcom.2019.10.033)

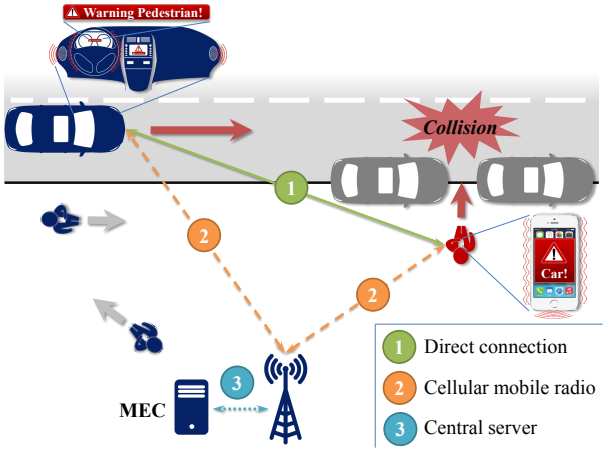


Figure 1: General concept of a VRU safety system.

is still unclear. Recognizing this gap along with the desire to leverage existing networks for V2X communications, we focus on studying the performance of cellular-based systems over the Uu interface and exploiting this approach for VRU safety applications/services. To be more specific, Car-to-Pedestrian (Car2P), a typical case of VRU systems, in which the safety for pedestrians is the main objective of our investigation in this paper.

In order to enable collision detection, pedestrians' smartphones and vehicles need to send beacon messages periodically to each other and after obtaining the beacons, a Collision Detection Algorithm (CDA) is executed. While these tasks are not a big deal for vehicles, they could be a significant obstacle when running on smartphones with limited resources. Especially, it would be a heavy burden on smartphones in terms of energy consumption and executing time when more information from other vehicles is needed and a more complex CDA is used. In this paper, we concentrate on addressing this bottleneck by considering the use of computation offloading using Multi-access Edge Computing (MEC). Applying this mechanism for smartphone applications, such as gaming, image/video processing, object/face recognition, or accelerated web browser has been shown to have a noticeable improvement in energy efficiency [14]; however, its applicability in the context of Car2P safety systems is still unclear.

The original idea of our approach was presented in our recent work [15], in which we proposed a dynamic selection of execution location (local or remote) for calculating pedestrian context information. Conceptually, the beacons transmitted between objects comprise some fundamental information like *current position*, *heading direction*, and *speed* [16, 17, 18, 19]. Some recent works [20, 3] suggested using additional context information to improve the accuracy of prediction results. Looking in more details, we can see that it is quite simple to calculate some information like position or direction directly from smartphone sensor data [21, 6]. But for others, e.g., motion states or activity, more extensive data handling like pre-processing, extraction of features, and training machine learning models are needed [22, 23, 20]. Depending on the amount of raw sensor data collected, an appropriate computational scheme is selected to save energy on smartphones.

In this paper, we extend and improve our proposed approach in [15], to support not only context information calculation but also the CDA. Generally, in cellular-based safety systems, context information computation and collision detection could be done either on a pedestrian smartphone itself or on a remote server. Deciding the strategy to not only prolong the battery life of the smartphone but also to meet the timing constraints in a Car2P safety system is a challenging task. It requires to take into consideration different parameters, such as energy consumption and processing time of local computation, network communication overhead, as well as real-time requirements of the system.

To provide insights on this adaptive approach, we carry out experimental and simulation studies on the performance of a Car2P system when applying both Local and Offload computational schemes on smartphones. In addition, energy consumption and latency of each scheme are investigated in the relationships with different parameters, such as the machine learning algorithm, the sensor sampling rate, the window size, and the sending interval of messages on the smartphone. We see the results of this paper as an important step towards energy-efficient VRU safety systems.

Our main contributions can be summarized as follows:

- We propose an adaptive computational approach for smartphones, which considers the possibility of offloading tasks in both data and service levels of Car2P safe applications.
- We measure and analyze the energy consumption and processing time of a lightweight machine learning application for determining pedestrian activities as well as a CDA for detecting potential collisions.
- We simulate a scenario of a Car2P system using the simulation framework Veins LTE to evaluate the end-to-end performance and scalability of our adaptive approach.

2. Related Work

The concept of a cooperative Car2P safety system has been described in many previous works [24, 5, 25, 1]. In particular, cellular-based safety systems have received much attention over the last years [5, 8, 18, 26]. Additionally, the communication performance of cellular in vehicular networks has been studied and compared to other wireless technologies, such as IEEE 802.11p-based DSRC [27, 9], which demonstrates the great potential of this technology in safety applications.

In general, information about the position, direction, and speed is used to predict potential collision between vehicles and pedestrians. However, in [21, 6, 28, 3] the

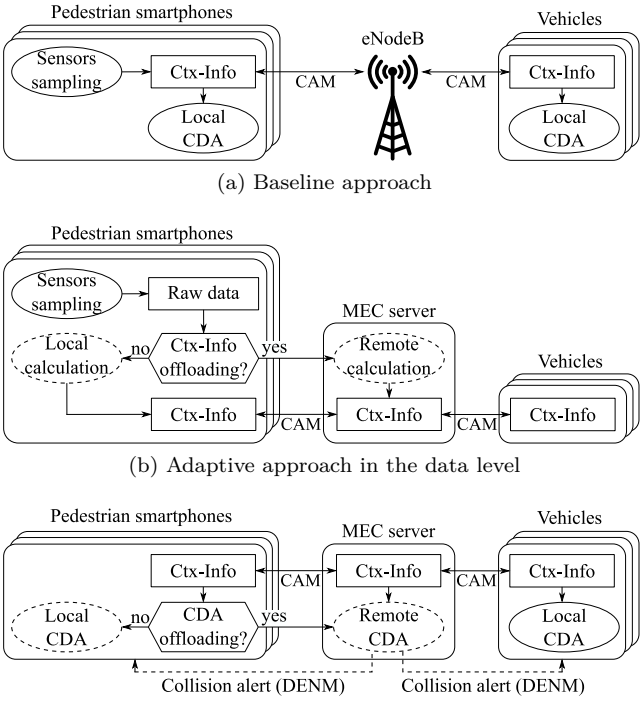
limitations of this information for collision detection have been pointed out. False detections are mainly due to the inaccuracy of position information and the frequent change of walking peoples' trajectories. Therefore, additional information like distraction level [6] (e.g., texting, watching a video or talking on the phone), VRU movements/activities [20, 3, 29] (e.g., stopping, walking, waiting, running, or crossing a curb), surrounding environment [16, 17] (e.g., indoor, outdoor, or in-vehicle), degree of risk [30] (e.g., approaching road, crossing road, or near vehicles), or weather condition, time of day, and age [31] have been suggested to be used as complementary knowledge, which helps to improve the accuracy of the detection or the efficiency of network communication. More specifically, the authors in [3] proposed an approach to use the pedestrian context *crossing a curb* and *stepping onto a road* to improve both the positioning accuracy and the collision detection probability. Moreover, in [29], the detection of cyclist movements, i.e., waiting or standing, are utilized to reduce the trajectory forecast error by selecting an appropriate forecasting model based on the detected movement. For the collision detection algorithm, many solutions have been introduced based on the context information exchanging between vehicles and pedestrians [32, 33, 3, 26]. It is noticeable that most of the research concentrated on the conceptual/architectural level without caring about the costs of calculating context information and running detection algorithms. The timing costs, as well as the influence of these calculations on the phone battery life, especially with higher complexity of the computation, are still unclear. Therefore, the feasibility of these algorithms is still an open question.

Realizing the restrictions of smartphones in terms of battery life and computation resources, several studies have focused on improving the energy efficiency of these devices when they are used in Car2P safety systems. One of the obvious ways to save energy on a smartphone is to reduce the amount of network traffic [18]. Depending on the risk level of pedestrians, an appropriate beaconing scheme for smartphones could be applied. In risk-free or low-risk situations, the smartphone can minimize data communication and vice versa in high-risk scenarios, a higher beacon frequency could be used. From the architectural perspective, it could be an option to save energy on a smartphone when local calculations are migrated to a remote server [18, 26]. The main challenge of this approach is to decide *when* to perform offloading in order to maximize the energy efficiency of the smartphone. The answer to this question depends on the actual conditions of the systems, networks, as well as the context at runtime. There have been many studies to solve the similar problem in other areas [14, 34]; however, to the best of our knowledge, there is currently no research on the performance of this strategy in the context of Car2P safety systems. Some existing studies [6, 16, 30] only consider the use cases in which the calculation is performed locally either on smartphones or vehicles, while some others [18, 26] suggested completely moving the execution of CDA to a central server.

To fill this gap, a study on a heterogeneous strategy is needed since only offloading or local computation is not always beneficial. In our recent work [15], we took the first step in solving this issue by examining the performance of the Offload scheme versus the Local scheme for safety context information calculation. In this paper, we further investigate the possibility of offloading computation in the service level, i.e., the CDA. The basis for such concepts is accurate measurement and estimation of energy consumption profiles for computation and communication tasks [35, 36]. Besides, in our system, we adopt MEC as a back-end server, like in [26], support forwarding Cooperative Awareness Messages (CAMs) to all concerned vehicles/pedestrians and offering context information calculation and collision detection services. This could later be extended to central-cloud [18] or micro-cloud solutions [37, 38].

3. System Architecture

Basically, our proposed Car2P safety system relies on the (centralized) architecture depicted in Figure 1. All communication within the system is established via LTE. Direct Car2P communication is beyond the scope of this paper. In the following, we explain our system architecture in which part of the information is processed locally and other parts are offloaded to the cloud supported by MEC.



(a) Baseline approach

(b) Adaptive approach in the data level

(c) Adaptive approach in the service level

Figure 2: MEC-based Collision Detection.

3.1. Baseline Approach

Figure 2a shows the baseline approach for a cellular-based Car2P safety system. As aforementioned, cellular communication is used as an alternative to D2D technologies, thus, all CAMs exchanged between vehicles and pedestrians are transmitted via an eNodeB (and the LTE core network). In general, the collision detection service works as follows: while a User Equipment (UE) (a vehicle or a pedestrian carrying his/her smartphone) is moving on the road, it frequently reports its existence to other UEs by sending CAMs, which contain its context information (Ctx-Info), to a base station (eNodeB) using the LTE interface. These CAMs will then be forwarded to other UEs in the same context. After receiving CAMs, UEs perform the CDA locally to predict the likelihood of crashes with other UEs and trigger collision detection actions if needed.

Taking a closer look at the smartphone side, it is obvious that local CDA is the most expensive computation task, especially when a high number of CAMs are received and the running period of the algorithm is small, i.e., the CDA is invoked more often. Besides, we can see that this approach requires context information to be calculated locally beforehand. Similar to the CDA, the smartphone also needs to run the calculation locally by itself. As discussed previously, it is not a problem for some basic information like position, speed, or direction, which just requires a few simple calculations. However, when more sophisticated and computationally intensive algorithms are needed to calculate context information like current activity or distraction level, this could consume a significant amount of energy of the battery and should be also taken into account in the optimization problem of energy consumption for the smartphone.

3.2. Adaptive MEC-based Collision Detection

Considering the limitations of the baseline approach, we propose an improvement to the architecture of the Car2P safety system, in which smartphones consider the possibility of computation offloading with the support of a MEC server. We suggest taking into consideration the applicability of this mechanism in both data level and service level.

3.2.1. Data Level

In the data level, smartphones decide whether or not to migrate context information calculation to the remote server. An overview of the data processing and communication flows of our adaptive approach in the data level is given in Figure 2b. In more detail, smartphones periodically collect raw data from different sensors. This data is then fed into a decision engine, which takes various parameters like current network condition and historical data as inputs and applies a certain logic to decide whether the computation for current context information should be offloaded to the MEC server or not. If the local computation is selected, the smartphone performs the calculation itself. The results are then used as inputs for the local CDA and along with other context information (e.g., positions) be encapsulated in a CAM to be sent to MEC for centralized processing. In the case of remote execution, the smartphone transmits raw sensor data to the MEC server, where context information is computed before being sent back to the smartphone and used as the inputs for the remote CDA.

In order to assess the performance of these two strategies, different parameters in terms of timing and energy consumption need to be taken into account. Let $E_{localCtx}$ and $T_{localCtx}$ be the energy and time consumed by calculating context information on the smartphone; E_{data} and T_{data} be the energy and time for transferring raw sensor data to the server; $T_{remoteCtx}$ be the time needed to calculate the context information on the server side; and $E_{resultCtx}$ and $T_{resultCtx}$ be the energy and time for receiving the calculated result from the server. The total delay time $T_{offloadCtx}$ since raw sensor data is ready until the calculated context information reaches the smartphone from the server can be given as

$$T_{offloadCtx} = T_{data} + T_{remoteCtx} + T_{resultCtx} \ . \quad (1)$$

In terms of time, the delay of the selected computation scheme, i.e., $T_{localCtx}$ for LOCAL and $T_{offloadCtx}$ for OFFLOAD, must not exceed a threshold T_{Ctx} (where T_{Ctx} is the maximum allowed delay for context information calculation). The offloading scheme is preferable, i.e., a better option to save energy, when

$$E_{data} + E_{resultCtx} < E_{local} \ . \quad (2)$$

Getting into more details, $E_{localCtx}$ and $T_{localCtx}$ are highly algorithm-dependent; E_{data} and T_{data} are closely related to the amount of raw sensor data collected; $E_{resultCtx}$, $T_{resultCtx}$, E_{data}, and T_{data} are influenced by the network status at the sending/receiving time; and $T_{remoteCtx}$ is determined by the computation power of the server.

3.2.2. Service Level

In the service level, smartphones consider the possibility of offloading the CDA. In Figure 2c, we show the processing flow of our adaptive approach in the service level. Similar to the data level, a smartphone also has two options: (1) taking the current context information and the CAMs received from other vehicles as inputs and perform the CDA locally; and (2) sending context information (encapsulated in a CAM) to a MEC server for processing and then waiting for a Decentralized Environmental Notification Message (DENM) in case potential collisions are detected. At the MEC server, all the data from different UEs is processed to predict the likelihood of crashes between UEs. If risky situations are detected, DENMs are then sent to all concerned UEs to trigger collision detection actions.

The condition for selecting operation schemes to save energy in the service level can be formulated as follows. Let $E_{localCDA}$ and $T_{localCDA}$ be the energy and time consumed by running the CDA on the smartphone; E_{CAM} and T_{CAM} be the energy and time for transferring a CAM to the server;

$T_{remoteCDA}$ be the time needed to execute the CDA on the server side; E_{DENM} and T_{DENM} be the energy and time for receiving a DENM from the server; and a threshold T_{CDA} be the maximum allowed end-to-end delay of the safety system. Running the CDA locally needs to satisfy the following timing constraint

$$T_{localCDA} \leq T_{CDA} \ . \tag{3}$$

Let $T_{offloadCDA}$ be the offloading time for the CDA execution, then it can be given as

$$T_{offloadCDA} = T_{CAM} + T_{remoteCDA} + T_{DENM} \ . \tag{4}$$

The *offloadCtx* scheme offers better energy efficiency when we have

$$E_{CAM} + E_{DENM} < E_{local} \tag{5}$$

$$\text{and} \qquad T_{offloadCDA} \leq T_{CDA} \ . \tag{6}$$

For local execution, $E_{localCDA}$ and $T_{localCDA}$ are related to the algorithm used and the number of CAMs to be processed. For offloading, $T_{remoteCDA}$ is dependent on the server. Other parameters including E_{CAM}, T_{CAM}, E_{DENM}, and T_{DENM} are associated with the network condition at runtime.

Generally, in both data and service levels, the LOCAL or OFFLOAD scheme has its own pros and cons. Deciding an appropriate operation scheme to improve the energy efficiency of smartphones while still guaranteeing the timeliness of messages received by pedestrian smartphones and vehicles requires a good understanding of the performance of context information computation and the CDA on both local and remote side as well as the overhead for data transfer between smartphones and the server.

3.3. Collision Detection Algorithm

In this section, we present the algorithm used in our system for detecting collisions between pedestrians and vehicles. On the smartphone side, the current position vector $\overrightarrow{p_0}$ and velocity vector $\overrightarrow{v}$ of the pedestrian are periodically updated. The smartphone also receives a set M of CAMs from nearby vehicles. Each $m \in M$ contains the current position vector $\overrightarrow{p_0^m}$ and the velocity vector $\overrightarrow{v^m}$ of the sending vehicle. With the assumption that the direction and speed of vehicles and pedestrians do not significantly change, their future positions over time are estimated as

$$\begin{aligned} \overrightarrow{p}(t) &\leftarrow \overrightarrow{p_0} + \overrightarrow{v}.t \ , \\ \overrightarrow{p^m}(t) &\leftarrow \overrightarrow{p_0^m} + \overrightarrow{v^m}.t \ . \end{aligned} \tag{7}$$

Basically, our CDA is based on the minimum distance between two objects [26]. In our context, the distance between the pedestrian and the vehicle is calculated as

$$\overrightarrow{d}(t) \leftarrow \overrightarrow{p}(t) - \overrightarrow{p^m}(t) \leftarrow (\overrightarrow{p_0} - \overrightarrow{p_0^m}) + (\overrightarrow{v} - \overrightarrow{v^m})t \ . \tag{8}$$

The square of the distance $D(t) = |\overrightarrow{d}(t)|^2$ is computed to simplify the calculation. The derivative is then used to calculated the time point t^* at which the distance $D(t)$ is minimum as

$$\begin{aligned} t^* &= t : \frac{d}{dt} D(t) = 0 \\ t^* &\leftarrow \frac{-(\overrightarrow{p_0} - \overrightarrow{p_0^m}).(\overrightarrow{v} - \overrightarrow{v^m})}{|\overrightarrow{v} - \overrightarrow{v^m}|^2} \ . \end{aligned} \tag{9}$$

If $t^* < 0$, the pedestrian and the vehicle are getting farther apart. If t^* is greater than a threshold T^* (where T^* is the time point at which the pedestrian/vehicle starts to pay attention to potential danger), the two objects are still in the safe areas to each other and there is no need for a warning. If $0 < t^* < T^*$, the minimum distance $d^* = \sqrt{D(t^*)}$ is computed. The sending of an alert message to the corresponding vehicle as well as a warning to pedestrian will be triggered if d^* is smaller than a certain threshold D^*. This process is performed for every CAM received from the vehicles.

If the collision detection service is carried out on the server side, current positions and velocities of all pedestrians and vehicles in the context are collected. The CDA is then applied for each pair of CAMs from vehicles and pedestrians. DENMs will be sent to the concerned vehicles/pedestrians if potential collisions are detected.

4. Experimental Study

In this section, we report on the experiments performed to evaluate the performance of our proposed adaptive approach in both data and service levels. As described in Section 3.2, energy consumption and processing time are the two main metrics, which directly affect the selection of operation schemes. In our experiments, we mainly focus on estimating the local computation costs as well as the energy consumed by network operations (transferring data to/from the server).

4.1. Experiment Setup

In the data level, we measure the energy consumption and time needed to perform context information calculation locally on a smartphone. We also compare the accuracy of different machine learning algorithm used for context information classification. In the service level, we conduct similar measurements but now for the CDA. We discuss our measurement results in relation to the different parameters: window length, sampling frequency, classifier, and the number of CAMs need to be processed. Based on the measurement results, we evaluate the energy efficiency of different computational combinations on data and service levels. All plots presented show average values together with 95 % confidence intervals.

We performed our experiments on a NEXUS 6 smartphone running Android V7.1.1. To measure the energy consumption of the smartphone while running the testing application, we implemented an Android background service to record the information related to the phone's

battery at runtime. Our service makes use of the smart battery interface of the smartphone, which provides measurement power/energy data with acceptable accuracy [36]. This is thanks to a Maxim MAX17050 battery fuel gauge equipped on the NEXUS 6 smartphone that offers precision measurements of current, voltage, and remaining charge. To measure the execution time, we compute the difference between the system time values recorded at the beginning and at the end of the calculation process. All logging data are stored in the local memory of the smartphone for offline statistics.

4.2. Context Information Calculation

4.2.1. Pedestrian Activity Recognition

As a prime example for context information calculation, we study the performance of the machine learning application for online determining pedestrian activities. Our application performs the calculation based on the smartphone sensor data from accelerometer and gyroscope, which detects whether a pedestrian is currently sitting, standing, walking, or running. Since the pedestrian's current activity has to be detected as quickly as possible so that the resulting data can be promptly given to the crash prediction application in the collision detection system, we implemented a lightweight version for Android smartphones.

A configuration for our algorithm is characterized by three parameters: window length, sensor sampling frequency, and classifier. We selected window length values of 0.1, 0.2, 0.5, 1.0, 1.5 and 2.0 s. For sampling sensor data, we chose frequencies of 10, 16, 32, 50 and 100 Hz. As shown in [39], a sampling frequency of 32 Hz is sufficient for tracking simple body movements based on the Shannon theorem. However, we deliberately chose higher frequencies as well in order to obtain enough data points when using smaller window sizes and to be able to detect faster or more complex activities, which may require a higher sampling rate. As part of our preprocessing, we extracted time domain features including mean, variance, minimum, and maximum. These features are sufficient for detecting simple movement activities and can be calculated with low computational effort [22, 40].

For our machine learning application, we used five different classifiers, which are frequently used for online activity recognition [22, 40, 41]: C4.5, Naive Bayes (NB), Random Forest (RF), JRip, and Support Vector Machine (SVM). In order to obtain reproducible results, we used the implementation from the Waikato Environment for Knowledge Analysis (WEKA) toolkit.[1] Each classifier model was trained for each pair of a certain sensor sampling frequency and window length beforehand and then selected according to the chosen configuration.

In order to investigate the energy and time efficiency of our machine learning algorithms and the offloading scheme for calculating pedestrian context information, we implemented three operation modes for our application on smartphones: (1) local pedestrian activity determination without updating the server (*localCtx*); (2) local pedestrian activity determination and then upload the result to the server (*localCtx++*); and (3) offloading pedestrian activity calculation to the server and receiving back the result (*offloadCtx*).

For the *localCtx++* and *offloadCtx* modes, we set up a simple server for handling packets sent from the smartphone. Data transmission between the smartphone and the server is performed using the UDP over an LTE connection. We put our smartphone at a fixed location with excellent LTE signal quality to minimize its effect on energy consumption of the smartphone during network communication. In order to improve the accuracy of the measurements, we disabled all unnecessary background services and the Wi-Fi interface on the smartphone during the experiments. We also kept the screen brightness of the phone at a fixed level. Finally, we repeated the experiment for each configuration 10 times to improve statistical confidence.

4.2.2. Classification Accuracy

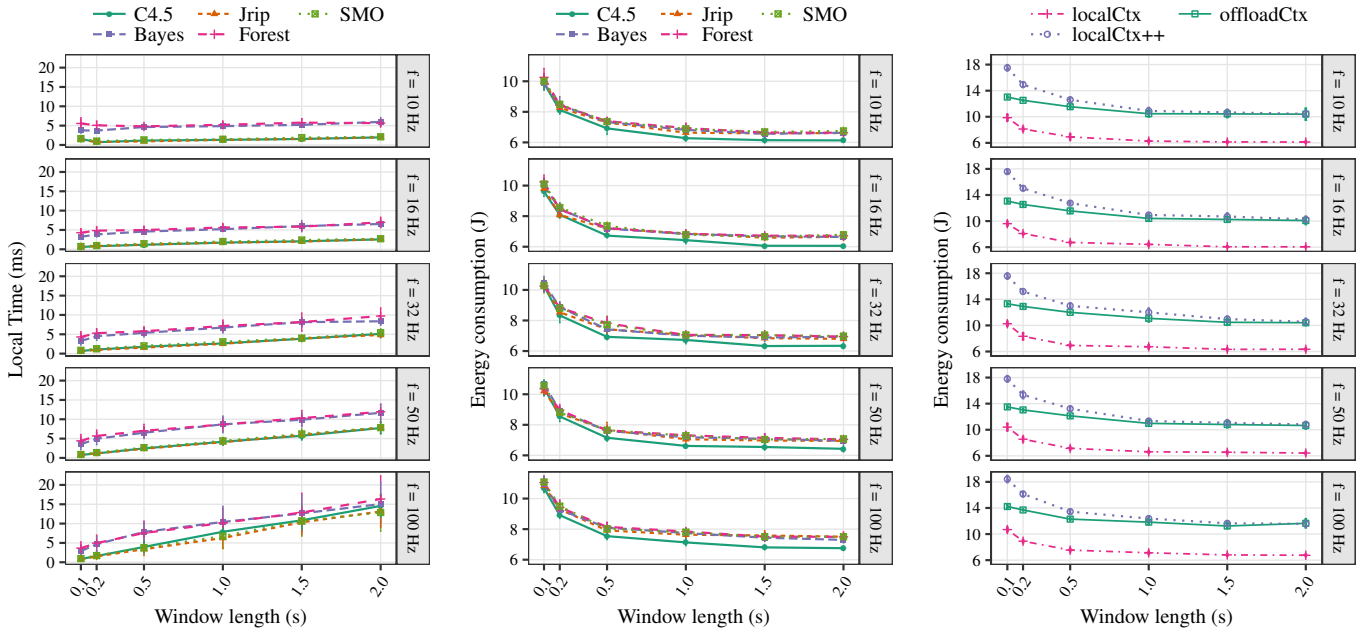
To estimate the accuracy of our trained models for the four chosen activities, we evaluated the performance of the five classification algorithms using the 10-fold cross-validation method. For all evaluations, we used a prerecorded dataset, which contains smartphone sensor data (accelerometer and gyroscope) captured while a participant performed four different activities, i.e., walking, running, standing, and sitting. This dataset was preprocessed for the respective configurations represented by a pair of selected frequency and window length. The obtained results are summarized in Table 1. The overall classification accuracies are greater than 90 % for all considered window lengths, frequencies, and classifiers, except the SVM classifier. The reason for these high accuracies is that only four continuous and simple activities have been considered for our investigation. In most cases, the RF classifier is found to produce the highest classification accuracies, while SVM shows the lowest accuracies, especially when using window lengths of 0.1 s.

However, since these results were obtained offline based on prerecorded sensor data, it should be noted that the classification results may differ when performing online classification in real time. It is due to unforeseen movements or changes in user behavior. Possible improvements towards classification accuracy, e.g., extracting additional features or changing parameters of the classification algorithms are left for future work. For the purpose of studying system performance, the current results are sufficient for our investigation.

4.2.3. Local Processing Time

Figure 3a shows our measurement results for the local processing time needed to perform classification using all considered machine learning algorithms for varying window lengths and sampling frequencies. The nearly linear

[1] `https://www.cs.waikato.ac.nz/ml/weka/`



(a) Local processing time. (b) Energy consumption of *localCtx* mode. (c) Energy consumption of C4.5 classifier.

Figure 3: Experimental results.

increase in processing time for higher frequencies and window lengths can be explained due to a higher amount of collected raw data, which increases the time needed to extract features.

Most notably, JRip, C4.5, and SVM exhibit lower processing times compared to NB and RF due to their lower computational complexity for classifying. In detail, C4.5 [42] trains a decision tree, in which classification is only performed on input features with the highest information entropy. JRip or RIPPERk [43], represents a propositional rule-based classification algorithm, which shows equal or lower processing times than C4.5, due to more aggressive repeated pruning that results in fewer rules to compare and thus, lower computation times in some cases. Finally, the SVM classifier [44] calculates the dot product between the input vector and the determined support vectors for all classes. The local processing time of this computation is comparable to JRip and C4.5 since our configuration of SVM only uses a linear kernel.

The NB classifier [45] on the other hand, assumes that all input values (features) are normally distributed and conditionally independent from each other. For every classification, the algorithm considers all input values and, therefore, shows an almost constant, linear increase in processing time. However, the average local time is higher compared to the other algorithms which only consider the most meaningful inputs. Throughout our investigations, the RF classifier showed similar or higher local processing times when compared to NB. This algorithm constructs random forests consisting of multiple random trees [46]. For classification of a given input vector, the result of the RF classifier is based on the output from all individual random trees. While the RF classifier shows slightly higher classification accuracies than the other algorithms (cf. Table 1), it also increases the local processing time.

4.2.4. Energy Consumption vs. Classifiers

To investigate the energy consumed by each classifier, we conduct the experiments for our application in the *localCtx* mode. We turned off the LTE interface during the experiments since there is no need for network communication in this mode. The measurement results are shown in Figure 3b.

First, it can be seen that the C4.5 classifier consumes the least energy, while the other algorithms consume slightly more energy, for all considered windows lengths and frequencies. An interesting observation here is that the energy consumed by the three algorithms is not directly correlated to the local processing time. For instance, although C4.5 requires a similar or in some cases higher local processing time than JRip, the energy consumption of C4.5 is slightly lower than JRip for all frequencies and window lengths, except for 0.1–0.2 s. The rest of the classification algorithms provide nearly similar results in terms of energy consumption.

Second, all classifiers show a decrease in energy consumption for higher window lengths. This is not surprising because smaller window sizes imply that the classification algorithm is called more often, and therefore consumes more energy. Moreover, rising sampling frequencies also increases the overall energy consumption.

WL [s]	Freq. [Hz]	Classification Accuracy [%] C4.5	JRip	NB	RF	SVM
0.1	10	93.04	93.23	91.88	95.91	50.97
	16	93.97	94.69	93.75	97.20	81.93
	32	95.04	96.11	93.55	97.65	83.38
	50	95.63	96.64	94.20	97.83	85.27
	100	96.46	96.46	94.00	97.89	86.30
0.2	10	94.64	94.72	95.00	97.14	84.17
	16	95.50	96.05	94.94	97.65	87.68
	32	95.90	96.42	95.42	97.84	90.01
	50	96.34	96.42	95.43	97.77	90.30
	100	96.74	96.82	96.02	97.97	90.26
0.5	10	94.35	95.44	95.73	97.42	90.87
	16	96.33	96.92	96.52	97.52	91.26
	32	97.22	96.92	97.22	97.71	91.45
	50	96.92	97.32	97.51	98.11	91.45
	100	96.42	96.92	97.12	97.51	91.55
1.0	10	95.44	95.63	96.03	98.21	91.67
	16	96.02	95.83	96.42	97.42	91.65
	32	95.23	95.63	96.42	97.81	91.25
	50	96.22	96.02	96.82	98.01	91.65
	100	96.02	96.22	97.02	97.81	91.85
1.5	10	93.75	92.56	95.83	96.13	90.48
	16	95.82	94.93	96.72	97.31	92.24
	32	96.12	95.82	95.82	97.61	91.04
	50	95.82	95.82	96.42	97.91	90.75
	100	94.33	93.73	95.82	97.31	90.75
2.0	10	96.03	94.05	96.03	96.83	90.08
	16	92.43	92.43	94.02	96.41	91.24
	32	93.23	96.02	95.22	97.21	91.63
	50	93.63	94.02	95.62	97.21	92.43
	100	95.22	96.02	95.62	96.81	91.63

Table 1: Classification results of all five algorithms using 10-fold cross-validation.

4.2.5. Energy Consumption vs. Modes

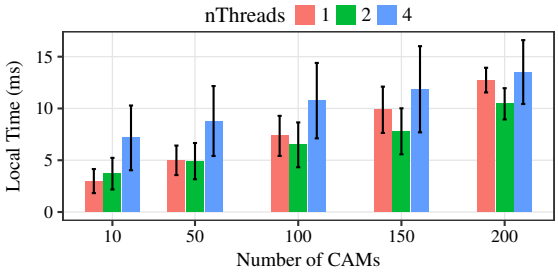
To evaluate and compare the energy efficiency between 3 operation modes mentioned in Section 4.2.1, we chose the C4.5 classifier, which yields classification accuracies of more than 92.43 % and the lowest computation time, as a representative to perform our experiments. In Figure 3c, we show the distribution of the energy measurements for this classifier with varying window lengths and sampling frequencies. Generally, the *localCtx* mode consumes much less energy than the others because, in this mode, the smartphone only performs the classification and no network operations are carried out. This fact proves that our machine learning algorithm is lightweight enough to be deployed for smartphones.

For the *localCtx++* mode, most of the energy consumed is due to the local computation of context information and only a small amount comes from uploading the result. On the other hand, the energy consumption in the *offloadCtx* lies in the transmission of raw sensor data. The amount of raw data is calculated as

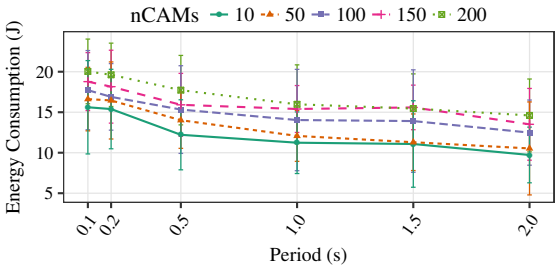
$$S = n_{sensor} \times f \times w \times m \times s_{type} \ , \tag{10}$$

where n_{sensor} is the number of sensors; f is the sensor sampling frequency; w is the window length; m is the number of values for each sensor data; and s_{type} is the size of the data type. For example, in our experiments, we were sampling data from accelerometer and gyroscope in 3 axes with 4 B each data value, thus, the amount of raw data can vary from 24–4800 B.

Basically, the *localCtx++* and *offloadCtx* modes show comparable energy consumption for large window lengths and a bit more with *localCtx++* mode for small ones. The difference in energy consumption between these two modes can be explained as follows: small window lengths (e.g., 0.1 s) mean the amount of raw sensor data collected is also small (e.g., 24 B), thus, the energy consumption for transmitting the raw data to the server in the *offloadCtx* mode is rather small compared to the quantity needed to locally extract the context information from this data in the *localCtx++*. For large raw data (e.g., 4800 B), the transmission energy is still only approximate the amount needed for local computation and result upload.



(a) Local time vs. number of CAMs and threads



(b) Local energy consumption vs. period

Figure 4: Local performance of collision detection algorithm.

4.3. Collision Detection Algorithm

We implemented the CDA as described in Section 3.3 on the smartphone. Like the context information calculation, we also supported two operation modes for our application: (1) locally execution of the CDA (*localCDA*) and (2) offloading the CDA to the server and receiving back DENMs if potential collisions exist (*offloadCDA*). We run our CDA with different number of CAMs, i.e., 10, 50, 100,

Simulation Parameter	Value
Simulated Area	1 km × 1 km
Layout	Intersection
Simulation time	30 s
Repetitions	30
LTE scheduler	MAXCI
Bandwidth	5 MHz (25 RBs)
UE transmission power	23 dBm
eNodeB transmission power	45 dBm
Number of vehicles	50
Beaconing interval (vehicles)	100 ms
Vehicle speed limit	50 km/h
Number of pedestrians	10, 50, 100, 150, and 200
Pedestrian speed limit	10 km/h
Window length/Period	0.1, 0.2, 0.5, 1.0, 1.5 and 2.0 s
Sensor sampling frequency	10, 16, 32, 50 and 100 Hz
Number of background UEs	10, 50, 100, 150, and 200
Background traffic	4 kB + uniform(−2 kB, 2 kB)
Background traffic interval	1 s + uniform(−0.5 s, 0.5 s)
CAM length	300 B
DENM length	500 B
resultCtx length	50 B

Table 2: Simulation Parameters.

150, and 200. The implement CDA was executed with 1, 2, and 4 threads and with the periods of 0.1, 0.2, 0.5, 1.0, 1.5 and 2.0 s. The measured local processing time and energy consumption are shown in Figure 4.

We can see that the overall processing time of our CDA is rather small (less than 15 ms) even with the high number of CAMs due to the low complexity of the algorithm. The processing time increases as the number of CAMs to be processed grows. Considering the performance when using three different number of running threads, it is shown that executing our CDA with 2 threads yields the best performance and with 4 threads produces the worst. In terms of energy consumption, our CDA consumes more for smaller periods and higher numbers of CAMs due to more frequent invocation and calculation.

5. Simulation Study

In this section, we evaluate the performance of our adaptive approach by means of network simulations. We conduct our study with the Veins LTE simulator [47], which is based on the popular Veins vehicular network simulation framework [48] building upon OMNeT++.[2] The road network and the movement patterns of vehicles and pedestrians are generated using SUMO.[3]

5.1. Simulation Scenario and Setup

The simulation scenario represents a simple context at an intersection between a highway and a road for pedestrians without traffic lights. The simulation parameters are listed in Table 2. We run the simulations with 50 vehicles and different numbers of pedestrians to investigate the behavior of our proposed approach. All vehicle and pedestrian modules are equipped with an LTE interface. We varied the speed of the vehicles from 0–50 km/h and the speed of the pedestrians from 0–10 km/h.

Beside modules generated from SUMO data, we deployed additional LTE UEs with random positions within the simulated area but do not require the safety service. These LTE UEs are used to generate background traffic, which makes the scenario more realistic. In general, background traffic could come from any common applications of mobile users like real-time video streaming, online gaming/music playing, or Internet surfing. The packet size and usage frequency of these applications may vary in an unpredictable manner. Therefore, in our simulations, we configured the background UEs to send and receive messages with random sizes and random intervals to/from the server. We also varied the number of these UEs as well while studying the influence of background traffic on system performance.

For simulating MEC, we installed a remote server, which is connected to the LTE eNodeB using a dedicated line. Therefore, the delay between the remote server and the base station is set to zero in our simulations. In the application layer, we implemented UDP-based modules for vehicles, background UEs, pedestrians, and the server. Vehicles send a CAM every 100 ms to the server and receive DENMs sent by the server. For pedestrians, we simulated two operation modes: offloading context information calculation (*offloadCtx*) and offloading CDA (*offloadCDA*). In the *offloadCtx* mode, the applications send messages, which contain raw sampling data from sensors, to the server, where the computation for pedestrian activities is performed. We varied the window length from 0.1–2.0 s and the sampling frequency from 10–100 Hz as in the experimental study. In our simulation, we assume that the application only samples the data from three sensors: accelerometer, gyroscope, and Global Navigation Satellite System (GNSS). Each data sample consists of three float values, i.e., (x, y, z) for accelerometer and gyroscope, (*latitude*, *longitude*, *elevation*) for GNSS. The size of data messages sent by pedestrian modules is calculated based on Equation (10). In the *offloadCDA* mode, the applications send CAMs with a fixed period to the server, where the CDA is executed. After sending a CAM, the application will wait for a DENM sent by the server. We also varied the period from 0.1–2.0 s.

For the server, we assumed that the server has unlimited computing resources, so the execution time for offloaded tasks can be neglected ($T_{remoteCtx} \approx 0$). First, the server responds to requests from background UEs. Similar to request messages, a reply also has a random length and random latency. Second, the server forwards CAMs from vehicles to pedestrians in the simulated area. And finally, the server offers the services (context information calculation and CDA) to pedestrians and vehicles. In the *offloadCtx* mode, after receiving a data message from a

[2] `http://www.omnetpp.org`
[3] `http://sumo.dlr.de`

pedestrian module, the server application will send back a result message of 50 B, which assumed to contain current activity information of the target pedestrian module. In the *offloadCDA* mode, after receiving a CAM from a pedestrian, the server will broadcast a DENM to all vehicles in the area as well as the origin pedestrian, which presumed to contain warnings about potential collisions. All simulations are repeated 30 times with an independent random seed for each run.

5.2. Simulation Results

We used two metrics to evaluate the performance of our proposed adaptive approach:

- End-to-end delay: is defined as the latency from the time a CAM or a data message containing raw sensor data is sent by a smartphone to the time the broadcast message from the server is received by a car or a pedestrian.
- Delivery rate: represents the reliability of the system, which is an essential factor in collision avoidance service. This metric is defined as the ratio of the total number of received messages to the total number of expected recipients.

Figures 5a and 5b show the changing of average end-to-end delay according to different parameters in both *offloadCtx* and *offloadCDA* modes, respectively. As the first observation, the average end-to-end delay in both schemes increases as the sensor sampling frequencies, window lengths, the density of pedestrians, and background LTE UEs grow. As for the number of pedestrians and background UEs, a higher density of these LTE UEs leads to higher connections to the server and higher network traffic, which can cause overload issues, and therefore increase packet latency.

In the *offloadCtx* mode, the size of messages transmitted by pedestrian modules increases rapidly relative to the grows of the window length and the sampling frequency, i.e., up to 7200 B for $w = 2.0$ s and $f = 100$ Hz. Messages with larger sizes require more time to send to the server, so clearly, the latency must be higher. On the contrary, in the conditions that the pedestrian and background UE densities are not too high (e.g., less than 150), the *offloadCtx* mode offers acceptable end-to-end delay (less than 100 ms) for short window lengths (e.g., less than 0.2 s) or frequency (e.g., less than 16 Hz). Looking at the *offloadCDA* mode, the end-to-end delay slightly increases as the period gets higher in our simulations.

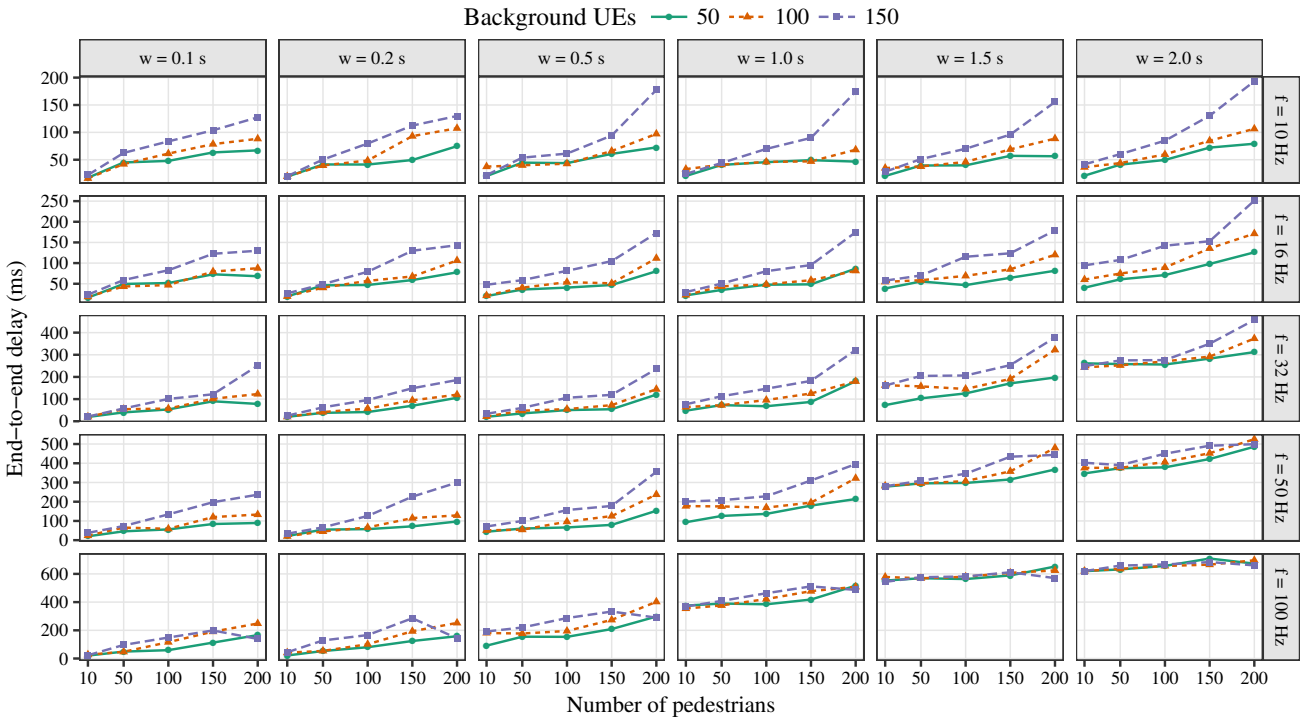
For the delivery rate, we can see in Figures 5c and 5d that this metric highly depends on the number of pedestrians and background UEs. The delivery rate drops rapidly when the number of pedestrians and background UEs grows due to higher traffic and connections to the server. In short, considering network communication, with centralized architecture, all CAMs are aggregated to the base station for the processing. There is a high probability of excessive connections, e.g., during rush hour with high user density, which puts a heavy burden on the base station and therefore could increase packet delay and decrease the delivery rate. In general, using different types of network connection, i.e., lowering traffic load on a single one, is a promising solution. Based on the actual status of each network connections, users can select the best option for data or services.

6. Discussion

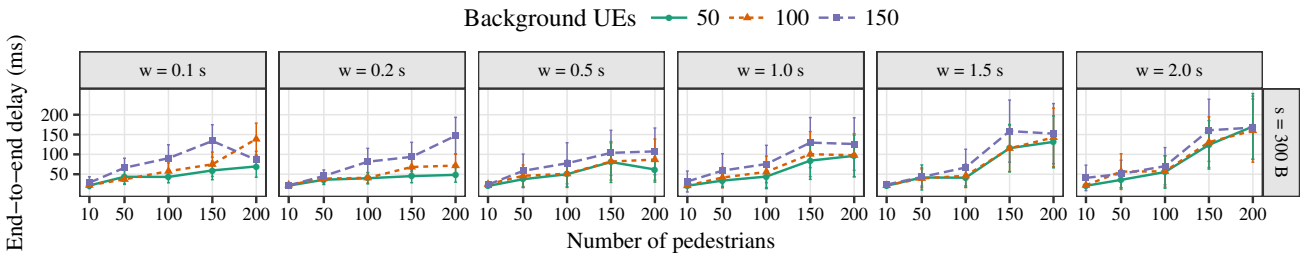
Based on the experimental and simulation results, we derived some very interesting insights, which will help to better optimize the Car2P safety systems in terms of energy efficiency for pedestrian smartphones and packet latency as well. As described in Section 3.2, context information calculation and CDA work together in a safety system for pedestrians as a two-phase process. The resulting context information of the first phase in the data level will be the input for the second phase, CDA, in the service level. First, it is evident in Figures 3a and 3c that the *localCtx* mode is very efficient in both terms of energy consumption and processing time. This mode is suitable when a pedestrian is in risk-free situations, where the smartphone only needs to self-check current context information without connecting to the server. When a pedestrian is in the circumstance that requires collision avoidance service, the smartphone could select one of the following combinations/schemes: (1) *localCtx-localCDA*; (2) *localCtx-offloadCDA*; (3) *offloadCtx-localCDA*; and (4) *offloadCtx-offloadCDA*.

Figure 6 depicts the energy measurements for all these computational combinations. For the first scheme, *localCtx-localCDA*, the smartphone completely performed the whole process locally, which consumes a significant amount of energy. On the other hand, the last scheme, *offloadCtx-offloadCDA*, decides to completely offload computational tasks to the server and waits for the results. In this scheme, the smartphone consumes the least energy comparing to other schemes since it only needs to send raw sensor data and positioning, speed, direction information (CAM) to the server without executing any computation. The heterogeneous scheme, *localCtx-offloadCDA*, has better energy efficiency than the fully local scheme, *offloadCtx-offloadCDA*, as it saves energy for the CDA. The *offloadCtx-localCDA* scheme is obviously the least effective in terms of energy consumption.

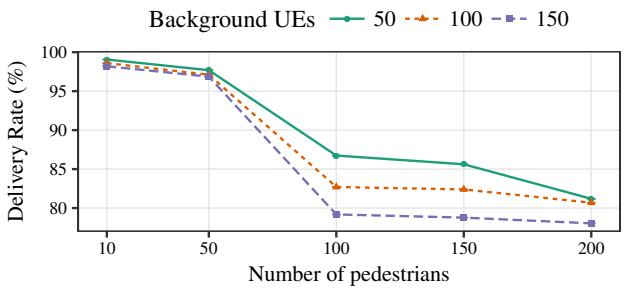
Even though the fully offloading scheme, *offloadCtx-offloadCDA*, exhibits the best energy efficiency; however, it is necessary to consider the timing performance as well since the timeliness of the messages in safety/warning systems is critical. Overall, the Local scheme offers low end-to-end delay in most cases, while as shown in Figures 5a and 5b, both *offloadCtx* and *offloadCDA* modes do not yield good latency performance as the LTE UE density grows. Therefore, a possible solution for such scenarios is to apply the Local scheme for the computational tasks. Since the safety of pedestrians has a higher priority than



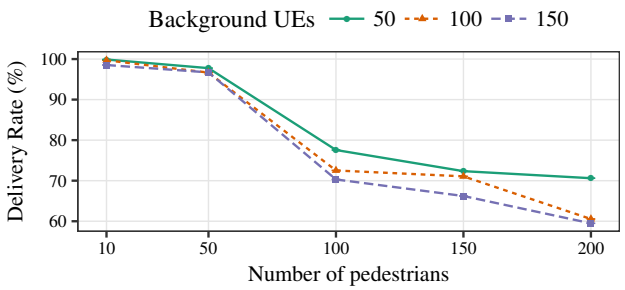
(a) Average end-to-end delay for offloading context information calculation (*offloadCtx*)



(b) Average end-to-end delay for offloading CDA (*offloadCDA*)



(c) Packet delivery rate for offloading context information calculation (*offloadCtx*)



(d) Packet delivery rate for offloading CDA (*offloadCDA*)

Figure 5: Simulation results.

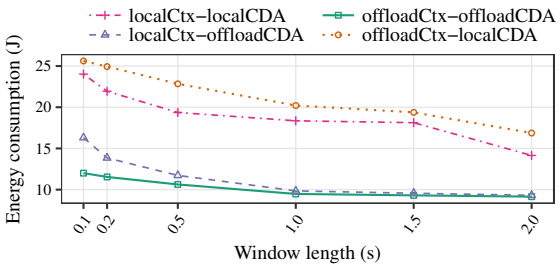


Figure 6: Energy consumption vs. operation schemes ($f = 50\,\text{Hz}$).

saving energy on the smartphone, even though the LOCAL scheme is not comparable to the OFFLOAD in terms of energy efficiency, the timeliness of the warning messages could be guaranteed. Another possible solution is to use a heterogeneous network, in which the smartphone could have different options of network interfaces for offloading tasks to the server.

7. Conclusion

In this paper, we studied options for Car2P communication with MEC support for improving the safety of VRUs. Building upon established C2C communication principles, we extend this concept to also integrate pedestrians into the system. The challenge here is that all computation and communication needs to be performed by a smartphone carried by the user. This poses both energy and latency issues given that dedicated activity detection algorithms need to be executed and the results need to be integrated into adequate CDAs. We proposed an offloading concept using MEC ideas for overcoming these limitations building upon a machine learning algorithm for real-time pedestrian activity recognition. We conducted both an experimental study of the energy consumption of our algorithms on smartphones as well as a simulation study of the LTE-based communication exploiting MEC servers. In conclusion, it can be said that offloading can substantially improve the situation and help updating the context information with reduced energy footprint.

References

[1] J. J. Anaya, P. Merdrignac, O. Shagdar, F. Nashashibi, J. E. Naranjo, Vehicle to pedestrian communications for protection of vulnerable road users, in: IEEE Intelligent Vehicles Symposium (IV 2014), IEEE, Dearborn, MI, 2014, pp. 1037–1042. doi:10.1109/IVS.2014.6856553.

[2] J. Heinovski, L. Stratmann, D. S. Buse, F. Klingler, M. Franke, M.-C. H. Oczko, C. Sommer, I. Scharlau, F. Dressler, Modeling Cycling Behavior to Improve Bicyclists' Safety at Intersections – A Networking Perspective, in: 20th IEEE International Symposium on a World of Wireless, Mobile and Multimedia Networks (WoWMoM 2019), IEEE, Washington, D.C., 2019. doi:10.1109/WoWMoM.2019.8793008.

[3] M. Bachmann, M. Morold, K. David, Improving smartphone based collision avoidance by using pedestrian context information, in: IEEE International Conference on Pervasive Computing and Communications Workshops (PerCom Workshops 2017), IEEE, Kona, HI, 2017, pp. 2–5. doi:10.1109/PERCOMW.2017.7917507.

[4] P. Sewalkar, J. Seitz, Vehicle-to-Pedestrian Communication for Vulnerable Road Users: Survey, Design Considerations, and Challenges, Sensors 19 (2) (2019) 358. doi:10.3390/s19020358.

[5] K. David, A. Flach, CAR-2-X and Pedestrian Safety, IEEE Vehicular Technology Magazine 5 (1) (2010) 70–76. doi:10.1109/MVT.2009.935536.

[6] X. Wu, R. Miucic, S. Yang, S. Al-Stouhi, J. Misener, S. Bai, W.-h. Chan, Cars Talk to Phones: A DSRC Based Vehicle-Pedestrian Safety System, in: 80th IEEE Vehicular Technology Conference (VTC2014-Fall), IEEE, Vancouver, Canada, 2014. doi:10.1109/VTCFall.2014.6965898.

[7] P. Choi, J. Gao, N. Ramanathan, M. Mao, S. Xu, C.-C. Boon, S. A. Fahmy, L.-S. Peh, A Case for Leveraging 802.11p for Direct Phone-to-Phone Communications, in: The International Symposium on Low Power Electronics and Design (ISLPED 2014), ACM, La Jolla, CA, 2014, pp. 207–212. doi:10.1145/2627369.2627644.

[8] L.-C. Tung, J. Mena, M. Gerla, C. Sommer, A Cluster Based Architecture for Intersection Collision Avoidance Using Heterogeneous Networks, in: 12th IFIP/IEEE Annual Mediterranean Ad Hoc Networking Workshop (Med-Hoc-Net 2013), IEEE, Ajaccio, France, 2013. doi:10.1109/MedHocNet.2013.6767414.

[9] Z. Xu, X. Li, X. Zhao, M. H. Zhang, Z. Wang, DSRC versus 4G-LTE for Connected Vehicle Applications: A Study on Field Experiments of Vehicular Communication Performance, Journal of Advanced Transportation 2017 (ID 2750452) (2017) 1–10. doi:10.1155/2017/2750452.

[10] R. Molina-Masegosa, J. Gozalvez, LTE-V for Sidelink 5G V2X Vehicular Communications: A New 5G Technology for Short-Range Vehicle-to-Everything Communications, IEEE Vehicular Technology Magazine 12 (4) (2017) 30–39. doi:10.1109/MVT.2017.2752798.

[11] S. Chen, J. Hu, Y. Shi, Y. Peng, J. Fang, R. Zhao, L. Zhao, Vehicle-to-Everything (V2X) Services Supported by LTE-based Systems and 5G, IEEE Communications Standards Magazine 1 (2) (2017) 70–76. doi:10.1109/MCOMSTD.2017.1700015.

[12] 5GAA, An assessment of LTE-V2X (PC5) and 802.11p direct communications technologies for improved road safety in the EU, Technical report, Automotive Association (Dec. 2017).

[13] G. Araniti, C. Campolo, M. Condoluci, A. Iera, A. Molinaro, LTE for Vehicular Networking: A Survey, IEEE Communications Magazine 51 (5) (2013) 148–157. doi:10.1109/MCOM.2013.6515060.

[14] P. Mach, Z. Becvar, Mobile Edge Computing: A Survey on Architecture and Computation Offloading, IEEE Communications Surveys & Tutorials 19 (3) (2017) 1628–1656. doi:10.1109/COMST.2017.2682318.

[15] Q.-H. Nguyen, M. Morold, K. David, F. Dressler, Adaptive Safety Context Information for Vulnerable Road Users with MEC Support, in: 15th IEEE/IFIP Conference on Wireless On demand Network Systems and Services (WONS 2019), IEEE, Wengen, Switzerland, 2019, pp. 28–35. doi:10.23919/WONS.2019.8795475.

[16] S. Tang, K. Saito, S. Obana, Transmission Control for Reliable Pedestrian-to-Vehicle Communication by Using Context of Pedestrians, in: IEEE International Conference on Vehicular Electronics and Safety (ICVES 2015), IEEE, Yokohama, Japan, 2015, pp. 41–47. doi:10.1109/ICVES.2015.7396891.

[17] A. Rostami, B. Cheng, H. Lu, J. B. Kenney, M. Gruteser, Performance and Channel Load Evaluation for Contextual Pedestrian-to-Vehicle Transmissions, in: 22nd ACM International Conference on Mobile Computing and Networking (MobiCom 2016), 1st ACM International Workshop on Smart, Autonomous, and Connected Vehicular Systems and Services (CarSys 2016), ACM, New York City, NY, 2016, pp. 22–29. doi:10.1145/2980100.2980103.

[18] M. Bagheri, M. Siekkinen, J. K. Nurminen, Cloud-Based Pedestrian Road-Safety with Situation-Adaptive Energy-Efficient Com-

munication, IEEE Intelligent Transportation Systems Magazine 8 (3) (2016) 45–62. doi:10.1109/MITS.2016.2573338.

[19] S. Loewen, F. Klingler, C. Sommer, F. Dressler, Backwards Compatible Extension of CAMs/DENMs for Improved Bike Safety on the Road, in: 9th IEEE Vehicular Networking Conference (VNC 2017), Poster Session, IEEE, Turin, Italy, 2017, pp. 43–44. doi:10.1109/VNC.2017.8275657.

[20] A. Jahn, K. David, S. Engel, 5G / LTE Based Protection of Vulnerable Road Users: Detection of Crossing a Curb, in: 82nd IEEE Vehicular Technology Conference (VTC2015-Fall), IEEE, Boston, MA, 2015, pp. 1–5. doi:10.1109/VTCFall.2015.7390782.

[21] S. Jain, C. Borgiattino, Y. Ren, M. Gruteser, Y. Chen, On the Limits of Positioning-based Pedestrian Risk Awareness, in: Workshop on Mobile Augmented Reality and Robotic Technology-based Systems (MARS 2014), ACM, Bretton Woods, NH, 2014, pp. 23–28. doi:10.1145/2609829.2609834.

[22] M. Shoaib, S. Bosch, O. D. Incel, H. Scholten, P. J. Havinga, A Survey of Online Activity Recognition Using Mobile Phones, Sensors 15 (1) (2015) 2059–2085. doi:10.3390/s150102059.

[23] C. A. Ronao, S.-B. Cho, Human activity recognition with smartphone sensors using deep learning neural networks, Expert Systems with Applications 59 (2016) 235–244. doi:10.1016/j.eswa.2016.04.032.

[24] C. Sugimoto, Y. Nakamura, T. Hashimoto, Development of Pedestrian-to-Vehicle Communication System Prototype for Pedestrian Safety Using both Wide-Area and Direct Communication, in: 22nd IEEE International Conference on Advanced Information Networking and Applications (AINA 2008), IEEE, Okinawa, Japan, 2008, pp. 64–69. doi:10.1109/AINA.2008.126.

[25] K. Dhondge, S. Song, B.-Y. Choi, H. Park, WiFiHonk: Smartphone-Based Beacon Stuffed WiFi Car2X-Communication System for Vulnerable Road User Safety, in: 79th IEEE Vehicular Technology Conference (VTC2014-Spring), IEEE, Seoul, South Korea, 2014, pp. 1–5. doi:10.1109/VTCSpring.2014.7023146.

[26] M. Malinverno, G. Avino, C. Casetti, C.-F. Chiasserini, F. Malandrino, S. Scarpina, Performance Analysis of C-V2I-Based Automotive Collision Avoidance, in: 19th IEEE International Symposium on a World of Wireless, Mobile and Multimedia Networks (WoWMoM 2018), IEEE, Chania, Greece, 2018, pp. 1–9. doi:10.1109/WoWMoM.2018.8449772.

[27] G. Cecchini, A. Bazzi, B. M. Masini, A. Zanella, Performance Comparison Between IEEE 802.11p and LTE-V2V In-coverage and Out-of-coverage for Cooperative Awareness, in: 9th IEEE Vehicular Networking Conference (VNC 2017), IEEE, Turin, Italy, 2017, pp. 109–114. doi:10.1109/VNC.2017.8275637.

[28] T. Datta, S. Jain, M. Gruteser, Towards City-Scale Smartphone Sensing of Potentially Unsafe Pedestrian Movements, in: 11th IEEE International Conference on Mobile Ad Hoc and Sensor Systems (MASS 2014), IEEE, Philadelphia, PA, 2014, pp. 663–667. doi:10.1109/MASS.2014.106.

[29] M. Bieshaar, S. Zernetsch, M. Depping, B. Sick, K. Doll, Cooperative starting intention detection of cyclists based on smart devices and infrastructure, in: 20th IEEE International Conference on Intelligent Transportation Systems (ITSC 2017), IEEE, Yokohama, Japan, 2017, pp. 1–8. doi:10.1109/ITSC.2017.8317691.

[30] A. Tahmasbi-Sarvestani, H. N. Mahjoub, Y. P. Fallah, E. Moradi-Pari, O. Abuchaar, Implementation and Evaluation of a Cooperative Vehicle-to-Pedestrian Safety Application, IEEE Intelligent Transportation Systems Magazine 9 (4) (2017) 62–75. doi:10.1109/MITS.2017.2743201.

[31] R. B. Zadeh, M. Ghatee, H. R. Eftekhari, Three-Phases Smartphone-Based Warning System to Protect Vulnerable Road Users Under Fuzzy Conditions, IEEE Transactions on Intelligent Transportation Systems 19 (7) (2018) 2086–2098. doi:10.1109/TITS.2017.2743709.

[32] G. R. de Campos, A. H. Runarsson, F. Granum, P. Falcone, K. Alenljung, Collision avoidance at intersections: A probabilistic threat-assessment and decision-making system for safety interventions, in: 17th IEEE International Conference on Intelligent Transportation Systems (ITSC 2014), IEEE, Qingdao, China, 2014, pp. 649–654. doi:10.1109/ITSC.2014.6957763.

[33] Q. Wu, L. C. Hui, C. Yeung, T. W. Chim, Early car collision prediction in VANET, in: 2015 International Conference on Connected Vehicles and Expo (ICCVE), IEEE, Shenzhen, China, 2015, pp. 94–99. doi:10.1109/ICCVE.2015.55.

[34] A. Bhattacharya, P. De, A survey of adaptation techniques in computation offloading, Journal of Network and Computer Applications 78 (C) (2017) 97–115. doi:10.1016/j.jnca.2016.10.023.

[35] M. Segata, B. Bloessl, C. Sommer, F. Dressler, Towards Energy Efficient Smart Phone Applications: Energy Models for Offloading Tasks into the Cloud, in: IEEE International Conference on Communications (ICC 2014), IEEE, Sydney, Australia, 2014, pp. 2394–2399. doi:10.1109/ICC.2014.6883681.

[36] Q.-H. Nguyen, J. Blobel, F. Dressler, Energy Consumption Measurements as a Basis for Computational Offloading for Android Smartphones, in: 14th IEEE/IFIP International Conference on Embedded and Ubiquitous Computing (EUC 2016), IEEE, Paris, France, 2016. doi:10.1109/CSE-EUC-DCABES.2016.157.

[37] F. Hagenauer, C. Sommer, T. Higuchi, O. Altintas, F. Dressler, Vehicular Micro Clouds as Virtual Edge Servers for Efficient Data Collection, in: 23rd ACM International Conference on Mobile Computing and Networking (MobiCom 2017), 2nd ACM International Workshop on Smart, Autonomous, and Connected Vehicular Systems and Services (CarSys 2017), ACM, Snowbird, UT, 2017, pp. 31–35. doi:10.1145/3131944.3133937.

[38] F. Hagenauer, T. Higuchi, O. Altintas, F. Dressler, Efficient Data Handling in Vehicular Micro Clouds, Elsevier Ad Hoc Networks 91 (2019) 101871. doi:10.1016/j.adhoc.2019.101871.

[39] G. Bieber, J. Voskamp, B. Urban, Activity Recognition for Everyday Life on Mobile Phones, in: 5th International Conference on Universal Access in Human-Computer Interaction (UAHCI 2009) - Intelligent and Ubiquitous Interaction Environments, Vol. LNCS 5615, Springer, San Diego, CA, 2009, pp. 289–296. doi:10.1007/978-3-642-02710-9_32.

[40] S. A. Hoseini-Tabatabaei, A. Gluhak, R. Tafazolli, A Survey on Smartphone-Based Systems for Opportunistic User Context Recognition, ACM Computing Surveys (CSUR) 45 (3) (2013) 27. doi:10.1145/2480741.2480744.

[41] D.-N. Lu, T.-T. Nguyen, T.-H. Nguyen, H.-N. Nguyen, Mobile Online Activity Recognition System Based on Smartphone Sensors, in: International Conference on Advances in Information and Communication Technology (ICTA 2016), Springer, Thái Nguyên, Vietnam, 2016, pp. 357–366. doi:10.1007/978-3-319-49073-1_39.

[42] J. R. Quinlan, C4.5: Programs for Machine Learning, Morgan Kaufmann, 1993.

[43] W. W. Cohen, Fast Effective Rule Induction, in: 12th International Conference on Machine Learning, Elsevier, Tahoe City, CA, 1995, pp. 115–123. doi:10.1016/B978-1-55860-377-6.50023-2.

[44] B. Schölkopf, C. J. Burges, A. J. Smola, Advances in Kernel Methods: Support Vector Learning, MIT Press, 1999.

[45] N. Friedman, D. Geiger, M. Goldszmidt, Bayesian Network Classifiers, Machine learning 29 (2-3) (1997) 131–163. doi:10.1023/A:1007465528199.

[46] L. Breiman, Random Forests, Machine learning 45 (1) (2001) 5–32. doi:10.1023/A:1010933404324.

[47] F. Hagenauer, F. Dressler, C. Sommer, A Simulator for Heterogeneous Vehicular Networks, in: 6th IEEE Vehicular Networking Conference (VNC 2014), Poster Session, IEEE, Paderborn, Germany, 2014, pp. 185–186. doi:10.1109/VNC.2014.7013339.

[48] C. Sommer, R. German, F. Dressler, Bidirectionally Coupled Network and Road Traffic Simulation for Improved IVC Analysis, IEEE Transactions on Mobile Computing 10 (1) (2011) 3–15. doi:10.1109/TMC.2010.133.