# Adaptive Safety Context Information for Vulnerable Road Users with MEC Support

Quang-Huy Nguyen*, Michel Morold‡, Klaus David‡, and Falko Dressler*
*Dept. of Computer Science and Heinz Nixdorf Institute, Paderborn University, Germany
‡Faculty of Electrical Engineering / Computer Science, University of Kassel, Germany
{nguyen,dressler}@ccs-labs.org
{michel.morold,klaus.david}@comtec.eecs.uni-kassel.de

***Abstract*—Cooperative Vulnerable Road User (VRU) collision avoidance aims at preventing potential accidents between VRUs and vehicles by exchanging context information. In this paper, we present a Multi-access Edge Computing (MEC)-based VRU safety system as an alternative to earlier purely ad-hoc communication-based ones, in which VRU smartphones utilize the cellular connection to frequently send context messages to a MEC server. However, in such safety systems, calculating context information on smartphones, which are already resource-restricted, could lead to reduced battery lifetime and, thus, to poor user experiences. To deal with this issue, we propose an adaptive approach for VRU context information calculation, which considers the use of computation offloading when needed in order to save energy while still ensuring timeliness. As a baseline, we use our machine learning application for determining pedestrian activities. Both experimental and simulation results suggest that it is worth to offload context information computation to the MEC when the updating interval or the sensor sampling frequency is low, i.e., the amount of raw data collected is small; otherwise, local execution is preferable. We see our results as a basis for designing more energy-efficiency calculation models for VRU safety systems.**

## I. Introduction

According to the latest report from the World Health Organization (WHO), 49 % of all road traffic accidents involve VRUs like pedestrians, bicyclists, or motor-cyclists [1]. Various approaches have already been introduced and are still being researched to reduce the number of these accidents. Existing solutions utilize vehicle sensors like cameras, laser scanner, and RADAR to detect VRUs and to avoid potential collisions [2]. However, those solutions only work in limited ranges and are usually dependent on a direct Line-Of-Sight (LOS) between vehicles and VRUs, which can be visually obstructed, for example by a parked vehicle. One idea to overcome this limitation is to track the movement and activities of a VRU beforehand for estimating whether a collision is possible. More recent research pursues a cooperative approach, in which VRUs are equipped with mobile devices like smartphones, which are able to exchange movement information with nearby vehicles. This approach offers many advantages; most importantly the VRUs can be detected even in Non-Line-Of-Sight (NLOS) scenarios.

Figure 1 depicts the general concept of a VRU safety system (extending our earlier work in [3]), in which VRUs carrying smartphones and vehicles, so called User Equipments (UEs), exchange their respective contextual information. Today, we can

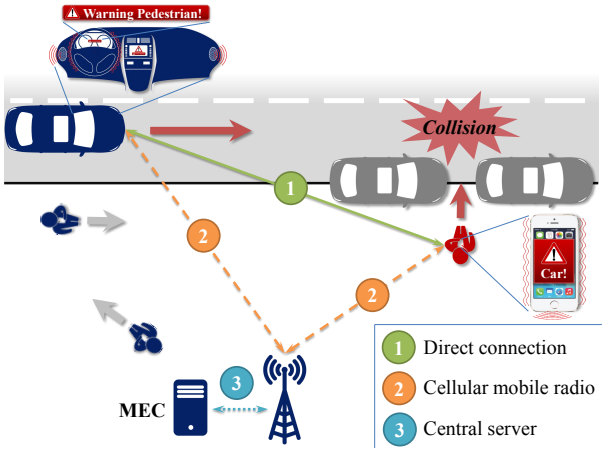



Figure 1: General concept of a VRU safety system.

distinguish two common communication architectures for VRU safety systems. One option is to use direct Device-to-Device (D2D) communication without the need of infrastructure and the alternative is an infrastructure-based approach. For D2D, existing technologies allow messages to be directly transferred from sender to receiver like WiFi Peer-to-Peer (P2P), Dedicated Short Range Communication (DSRC), or the more recent LTE Direct. DSRC has been commonly assumed for Car-to-Car (C2C) communication, but it is still in the trial phase, particularly when considering smartphones [4], [5]. Similarly, LTE Direct is considered having a high potential in Vehicle-to-Everything (V2X) [6]; yet, integration with smartphones is not yet clear. So, at the moment, the most promising approach is to rely on infrastructure. Considering cellular communication, UEs participate in a larger network of a mobile communication system. Following the current evolution from LTE to 5G, the technology has many advantages for VRU safety systems – and it is also considered for V2X communication [6], [7].

For collision prediction, beacons transmitted by VRU smartphones should comprise some basic information like *current position*, *heading direction*, and *speed* [8]–[11], and maybe some additional ones like current distraction and motion states [4], or weather condition, time of day, and age [12], which could help improving the prediction of impending collisions. While some information (e.g., position or direction) can be obtained directly or through simple calculations from

smartphone sensor data, others (e.g., motion states) require more extensive data handling like pre-processing, extraction of features, and training machine learning models [13], [14].

However, restricted resources are one of the biggest issues facing smartphones. The more context information and complex algorithms are used, the greater computing burden on smartphones and, therefore, the shorter their battery lifetime. To address this bottleneck, in addition to efforts to lighten the applications, we propose an adaptive approach for calculating VRU context information. The core idea of our approach is to try to reduce local execution as much as possible by offloading tasks to a remote server. Besides, we suggest taking the advantages of MEC, i.e., ultra-low latency, high bandwidth, and real-time access to radio network information, for road-safety applications.

This architecture was shown to have noticeable benefits in many areas, such as gaming, image/video processing, object/face recognition, or web accelerated browser [15]; however, its applicability in the context of VRU safety systems is still an open question. We investigate this issue by studying various computational schemes on the smartphone with regard to energy consumption and latency. Moreover, these costs also depend on many other parameters, such as the machine learning algorithm, the sensor sampling rate, the window size, and the sending interval of messages on the smartphone, which are also taken into account in our measurements. We see the results of this paper as an important step towards energy-efficient VRU safety systems. To be more specific, in this paper, we focus on Car-to-Pedestrian (Car2P) systems, a typical case of VRU systems, in which the safety for pedestrians is the main objective of our investigation.

Our main contributions can be summarized as follows:

- We propose an adaptive approach for calculating VRU context information used by collision avoidance services.
- We measure and analyze the energy consumption and processing time of a lightweight machine learning application for determining pedestrian activities.
- We integrate the experimental results into the simulation framework Veins LTE to evaluate the end-to-end performance and scalability of our adaptive approach.

## II. Related Work

Research towards collaborative, smartphone-based collision avoidance has received much attention over the last two decades. In [3], [16]–[18], the architectures for Car2P safety system have been proposed, in which pedestrians are equipped with smartphones that enable the exchange of necessary information with nearby vehicles. Generally, the collision risk is anticipated using position and movement information obtained from Global Positioning System (GPS) and sensors on the phones.

Some previous works [4], [19], [20] pointed out the disadvantages of positioning-based information for detecting pedestrian risk due to the positioning error and the frequent changes of walking people trajectories. Therefore, beside the movement vector, the authors suggested to incorporate a pedestrian activity identifier and a distraction monitor on smartphones to provide additional information for collision prediction. Depending on the current motion of pedestrians, e.g., stopping, walking or running, and distracting activities, e.g., texting, listening to music, or talking on the phone, safety algorithms or warning strategies could be adjusted to enhance their accuracy and reliability. However, there was no evaluation of the influence of these modules on energy consumption of smartphones, so the feasibility of these algorithms was unclear.

More recent publications [8], [9], [21] considered the use of contextual information to improve the efficiency of network communication. Basically, this context-aware approach requires smartphones to integrate additional detection modules for motion state, surrounding environment (indoor, outdoor, or in-vehicle), and degree of risk (approaching road, crossing road, or near vehicles). Smartphones of pedestrians, who are in higher risk situations, are given higher priority on the channel to reliably send their messages to the vehicles in the same context. Besides, it was also suggested that the smartphones could turn off network communication or at least reduce the transmission frequency when the pedestrian is stationary or not in danger. This way, the traffic load could be significantly reduced and thus, improve the network performance. Again, it was not mentioned in the paper how each mechanism affects the battery life of smartphones.

The work presented in [10] is most closely related to ours. Here, the authors evaluated the energy consumption of smartphones when applying different beaconing schemes according to the risk level of pedestrians. Additionally, the limitation of battery lifetime has been tackled from architectural perspective. Unlike other systems where predicting accidents is performed on smartphones or vehicles, a cloud-based server is employed for the calculation. Cellular connections (3G and LTE) are used for the communications between cars/pedestrians and the server. This method releases the computational burden on smartphones, thus, saving more energy. Using the similar approach, in this paper, we further improve the energy efficiency of smartphones by considering the offloading problem at the data level, i.e., context information calculation. We also examine a more general problem, in which both local and offload schemes are investigated, since only offloading to servers is not always beneficial. The basis for such concepts is an accurate measurement and estimation of energy consumption profiles for computation and communication tasks [22], [23].

Regarding simulation-based performance evaluation, most existing papers focused on safety systems for vehicles [24], [25]. Some recent works developed simulation models for pedestrians [9], [10]. The general idea is to combine a discrete-even simulator (OMNeT++ or ns-3) with the Simulation of Urban MObility (SUMO) framework. In the same manner, in this paper, we conduct a simulation study with the Veins LTE framework [26], [27]. The road network and the movement patterns of vehicles and pedestrians are generated by SUMO. To our best knowledge, our work is the first simulation study integrating person objects into the Veins simulator.

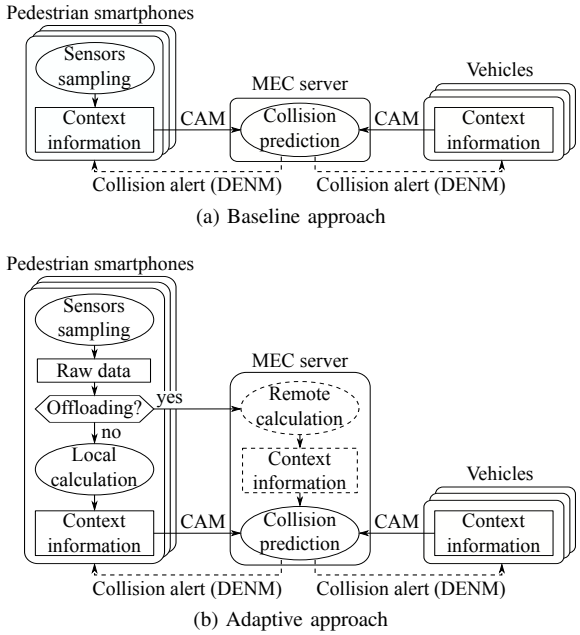



Figure 2: MEC-based Collision Avoidance.

## III. System Architecture

Basically, our proposed Car2P safety system relies on the centralized architecture, where MEC as a back-end server, offers data processing and computation for potential accidents. The overall architecture is depicted in Figure 1. Communication within the system is established via LTE. Direct Car2P communication is beyond the scope of this paper.

### A. MEC-based Collision Avoidance

Figure 2a shows the baseline approach for a MEC-based Car2P safety system. In general, the collision avoidance service works as follows: while a UE (a vehicle or a pedestrian carrying his/her smartphone) is moving on the road, it frequently sends Cooperative Awareness Messages (CAMs), which contain its context information, to a base station (eNodeB) using the LTE interface to report its existence. All the data from different UEs is processed by a MEC server deployed at the base station to predict the likelihood of crashes between UEs. If risky situations are detected, Decentralized Environmental Notification Messages (DENMs) are then sent to all concerned UEs to trigger collision avoidance actions. To some extent, performing prediction at the server rather than at UEs helps to cope with the limited resource issue of smartphones.

Taking a closer look at the smartphone side, we can see that this approach requires context information to be calculated locally beforehand. This could be a drawback for smartphones in terms of energy consumption, especially when more sophisticated and computationally intensive algorithms are needed to calculate context information.

### B. Adaptive Approach for Context Information Calculation

We therefore propose an improvement to the baseline approach, in which smartphones consider the possibility of offloading pedestrian context information calculation to the MEC server. An overview of the data processing and communication flows of our adaptive approach 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 MEC server or not. If local computation is selected, the smartphone performs the calculation itself and then uploads the results to MEC. In the case of remote execution, the smartphone transmits raw sensor data to the server and here, context information is computed before being used as the inputs for collision prediction.

The condition for selecting operation schemes to save energy can be formulated as follows. Let $E_{local}$ and $T_{local}$ be the energy and time consumed by calculating context information on the smartphone; $E_{CAM}$ and $T_{CAM}$ be the energy and time for sending a CAM to the server; $E_{data}$ and $T_{data}$ be the energy and time for transferring raw sensor data to the server; $T_{DENM}$ be the time for sending a DENM from the server to an UE; and $L$ be the maximum allowed end-to-end delay of the safety system. We assume that the server has unlimited computing resources, i.e., the remote execution time can be neglected. First, it should be noted that an algorithm for context information calculation is only acceptable if it satisfies the following timing constraint

$$T_{local} + T_{CAM} + T_{DENM} \leq L \ . \tag{1}$$

The offloading scheme is then preferable, i.e., a better option to save energy, when we have

$$E_{data} < E_{local} + E_{CAM} \tag{2}$$

$$\text{and} \qquad T_{data} + T_{DENM} \leq L \ . \tag{3}$$

Getting into more details, $E_{data}$ and $T_{data}$ are closely related to the amount of raw data collected, while $E_{local}$ and $T_{local}$ are highly dependent on the algorithm used. Generally, each operation scheme (local or offload) has its own pros and cons. To make appropriate offloading decisions to improve the energy efficiency of smartphones while still assure the timeliness of messages received by the MEC server and vehicles, it is essential to have a good understanding of the performance of context information computation on smartphones as well as the overhead for data transfer to/from the server.

As a prime example, we study the performance of the machine learning application for determining pedestrian activities. The general algorithm of our application can be described as follows: the program first collects raw data from the smartphone accelerometer and gyroscope for a chosen window length and sampling frequency. In the preprocessing phase, a sliding window approach is applied on the raw data for features extraction. We chose time domain features including mean, variance, minimum, and maximum, which yielded good results in prior works [13] and can be calculated with low computational effort. The features extracted from the sensor data are fed to a classifier to discriminate pedestrian activities.

In order to investigate the energy and time efficiency of our machine learning algorithms and adaptive approach for calculating pedestrian context information, we study the following operation schemes on smartphones:

1) LOCAL: the current pedestrian activity is determined locally on the smartphone without updating the server;
2) LOCAL++: the current pedestrian activity is determined locally on the smartphone and the result together with other context information (e.g., positions) are encapsulated in a CAM to be sent to the server;
3) OFFLOAD: raw sensor data is collected for a specific duration (window length) and sent to the server, where the classification for pedestrian activity is performed;
4) STREAM: this is a special case of the OFFLOAD scheme, in which every sample of raw sensor data is immediately streamed to the server for the classification without waiting for enough data for the selected window length.

## IV. EXPERIMENTAL STUDY

In this section, we present our experiments to measure the energy consumption and processing time of the machine learning application for determining pedestrian activities running on smartphones. We discuss our measurement results in relation to the operation schemes presented in Section III-B as well as the following parameters: window length, sampling frequency, and classifier. All the plots presented show the average values with 95 % confidence intervals.

### A. Experiment Setup

Our experiments were performed on a NEXUS 6 smartphone running Android v7.1.1. To measure energy consumption of a smartphone, two possible options could be considered, i.e., using external devices and self-measurement using the Smart Battery Interface [22], [23]. In our case, the NEXUS 6 smartphone integrates a Maxim MAX17050 battery fuel gauge, which provides measurements of instantaneous current and remaining charge. The power/energy consumption of the smartphone can be determined by this in-system chip with acceptable accuracy and there is no need to connect external devices to the phone [23]. To make use of this functionality, we implemented an Android background service to record the information related to the power properties of the battery at runtime. 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.

For online pedestrian activity recognition, our application performs the calculation based on smartphone sensor data, 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 avoidance 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 0.2, 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 [28], 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. Our application supports three different classifiers, namely C4.5/J48, Naive Bayes, and K-Nearest Neighbors (KNN). For each classifier, we used the implementation from the Waikato Environment for Knowledge Analysis (WEKA) toolkit.[1] We trained each classifier for each pair of a certain sensor sampling frequency and window length beforehand and selected the appropriate classifier model according to the chosen configurations.

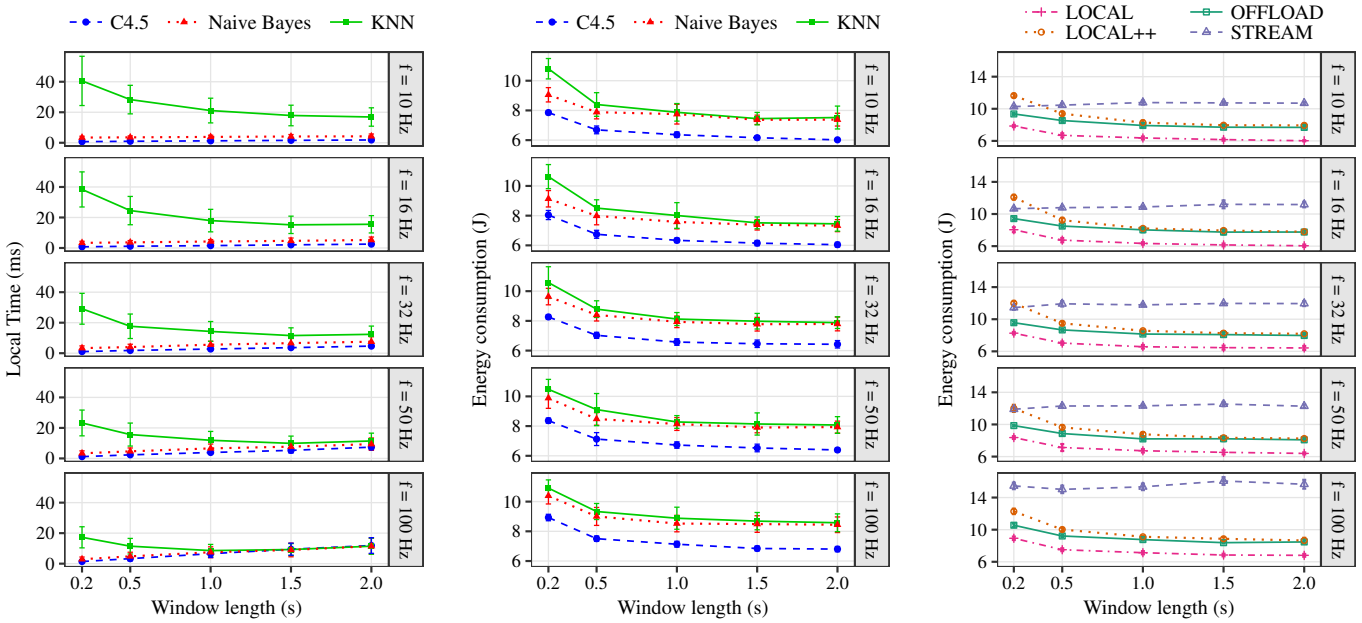
For experimentation, we set up a simple server for receiving packets from the smartphone. Data transmission between the smartphone and the server is performed using the UDP over an LTE connection. We put our smartphone at a fixed location with excellent LTE signal quality to minimize its effect on energy consumption of the smartphone during network communication. In order to improve the accuracy of the measurements, we disabled all unnecessary background services and the WiFi 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.

### B. Local Processing Time

Figure 3a shows our measurement results of the local processing time needed to perform one classification using C4.5, Naive Bayes, and KNN for varying window lengths and sampling frequencies. Most notably, the KNN classifier needs a higher processing time in almost cases when compared to the other algorithms due to its higher computational complexity for classifying. In details, the KNN classifier uses a lazy learning method, which only requires a little training time (offline) but is rather computationally expensive in the testing phase. Every new instance obtained has to be compared with all individual data points of the training set in order to determine the most probable class (i.e., activity). Naive Bayes, on the other hand, represents a lightweight classifier, which shows a near constant evaluation time due to the assumption that all input values (i.e., features) are normally distributed and conditionally independent from each other. C4.5 trains a decision tree, in which classification is only performed on input features with the highest information entropy.

Besides, for higher frequencies and window lengths, the KNN classifier shows a decrease in processing time, while the C4.5 and Naive Bayes exhibit a slight linear increase in local time with the increasing these two parameters. This can

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




(a) Local processing time. (b) Energy consumption in the LOCAL scheme. (c) C4.5 energy consumption vs. operation schemes.

Figure 3: Experimental results.

be explained due to the fact that both Naive Bayes and C4.5 classifiers have more constant computational complexity for different window lengths and frequencies. The increase in total processing time of both algorithms is because of the increasing amount of collected raw data, i.e., more time is needed to extract features. This is not the case with the KNN classifier since its processing time depends on the $K$ variable, whose value is picked to get the best possible fit for each data set corresponding to the pair of window length and frequency. In our application, for smaller window lengths and frequencies, the KNN has to compare more data points to make classification decisions, which leads to higher processing times.

### C. Energy Consumption vs. Classifiers

To investigate the energy consumed by each classifier, we conduct the experiments for our application in LOCAL scheme. We turned off the LTE interface during the experiments since there is no need for network communication in this scheme. The measurement results are given in Figure 3b. First, it can be seen that the C4.5 classifier consumes the least energy, followed by the Naive Bayes and KNN. This is because the C4.5 algorithm spends the least time on computation while the KNN requires the most. However, an interesting observation here is that the energy consumed by the three algorithms is not directly correlated to the local processing time needed. The KNN classifier, for example, consumes only slightly more energy than the Naive Bayes classifier, despite taking much more time for local computation.

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. Considering the relationship between sampling frequencies and energy consumption, we can see that there are slightly increases in energy consumed by C4.5 and Naive Bayes classifiers, while the amount consumed by KNN does not change much.

### D. Energy Consumption vs. Operation Schemes

To evaluate and compare the energy efficiency between operation schemes, we chose the C4.5 classifier, which yields classification accuracies of more than 95.13 % 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 in all 4 operations schemes with varying window lengths and sampling frequency. Generally, the LOCAL scheme consumes much less energy than the others because in this scheme, the smartphone only performs the classification and no network operations is carried out. This fact proves that our machine learning algorithm is lightweight enough to be deployed for smartphones. The LOCAL++ and OFFLOAD schemes show a comparable energy consumption for large window lengths and a bit more with LOCAL++ scheme for small ones. The difference in energy consumption between these two schemes can be explained as follows: small window lengths mean the amount of raw sensor data collected is also small, and thus, the energy consumption for uploading this data to the server is rather slight compared to the quantity for local computation, and vice versa. The LOCAL++ scheme, therefore, needs more energy than the OFFLOAD scheme for small window lengths.

For the STREAM scheme, the energy consumption is much higher than the other schemes in most cases due to its high frequency of sending messages to the server. Moreover, all operation schemes except STREAM exhibit a decrease in

Table I: Simulation Parameters.

| Simulation Parameter | Value |
|---|---|
| Simulated Area | 1 km × 1 km |
| Layout | Intersection |
| Simulation time | 60 s |
| Repetitions | 30 |
| Other LTE UEs | 25, 50, 100, 150, and 200 |
| Background LTE traffic | 1 kB + uniform(−0.5 kB, 0.5 kB) |
| Background LTE traffic interval | 0.5 s + uniform(−0.25 s, 0.25 s) |
| Window length | 0.2, 0.5, 1.0, 1.5 and 2.0 s |
| Sensor sampling frequency | 10, 16, 32, 50 and 100 Hz |
| **LTE Parameter** | **Value** |
| Bandwidth | 5 MHz (25 RBs) |
| LTE scheduler | MAXCI |
| UE transmission power | 23 dBm |
| eNodeB transmission power | 45 dBm |

energy for higher window lengths. This is because the data sending frequency in the STREAM scheme is determined by the sampling frequency, while the invocation frequency in other schemes is decided by the window length.

## V. Simulation Study

This section describes our performance evaluation for the proposed Car2P safety system using the Veins LTE vehicular network simulation framework [27] building upon OMNeT++. The road network and the movement patterns of vehicles and pedestrians are generated using SUMO.

### A. Pedestrian Module in Veins

To our best knowledge, our work is the first simulation study integrating person objects in the Veins simulator. In SUMO, we can define a person[2] as a `vType` object with `vClass=pedestrian`. It is also possible to define different movements for a given person including *ride*, *walk*, and *stop*. In Veins, we created a new node type, namely `Pedestrian`, for person objects, which is defined as an OMNeT++ module. Similar to the existing *vehicle* modules, e.g., `HeterogeneousCar`, to retrieve information of *person* objects from SUMO, we implemented a set of TraCI commands[3] for this type of module in Veins, which is managed by `TraCIScenarioManager`.

### B. 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. We varied the speed of the vehicles from 0 km/h up to 50 km/h and the speed of the pedestrians from 0 km/h (stop) up to 15 km/h (sprinting).

We assume that our scenario area is covered completely by the LTE eNodeB. All vehicles and pedestrians are equipped with an LTE interface. 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 both pedestrian smartphones and the server. To simulate the background traffic in the network, we deployed a number of LTE UEs with random positions within the simulated area, sending messages with random sizes and random intervals to the server. We varied window length from 0.2–2.0 s and the sampling frequency from 10–100 Hz as in the experimental study. Other simulation parameters are summarized in Table I.

We performed simulation experiments for these schemes: LOCAL++, OFFLOAD, and STREAM. In the LOCAL++ scheme, we assume that the current activity of a pedestrian is determined locally using the classifiers mentioned in the previous sections and the result is then encapsulated into a CAM of 300 B to be sent to the central sever. We simulated the timing of local computation by delaying the operations of sending CAMs by the time interval, retrieved from our measurement results of local processing time presented in Section IV-B.

In the OFFLOAD scheme, the application sends messages, which contain raw sampling data from sensors, to the server, where the computation for pedestrian activities and the potential of collision is performed. The size of messages sent by person objects is calculated as

$$S_{message} = n_{sensor} \times f \times w \times m \times s_{type} \ , \qquad (4)$$

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 data type. 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 three float values, i.e., ($x$, $y$, $z$) for accelerometer and gyroscope, ($latitude$, $longitude$, $elevation$) for GNSS. In the STREAM scheme, the sending interval is decided by the sampling frequency.

Assuming that the server has unlimited computing resources, the execution time for offloaded tasks can be neglected. For all schemes, when receiving a CAMs or a raw data message from a pedestrian, the server immediately broadcasts a DENM to all nodes in the network, which assumed to contain current context information or warnings about potential collisions. For our simulations, we set the size of broadcast messages sent by the server of 500 B. All simulations are repeated 30 times with an independent random seed for each run.

### C. Simulation Results

We used end-to-end delay (latency) metric to evaluate the performance of our proposed system. This metric is defined as the delay from the time the raw sensor data is available on smartphones to the time the broadcast message from server is received by a car or a pedestrian. Figure 4 shows the changing of average end-to-end-delay according to sensor sampling frequencies, window lengths, operation schemes, and the density of other LTE users.

As the first observation, the average end-to-end delays in the LOCAL++ and STREAM schemes are less affected by the window length, sampling frequency, as well as the density of LTE users. The LOCAL++ scheme shows an average delay

[2]http://sumo.dlr.de/wiki/Specification/Persons

[3]http://sumo.dlr.de/wiki/TraCI/Person_Value_Retrieval

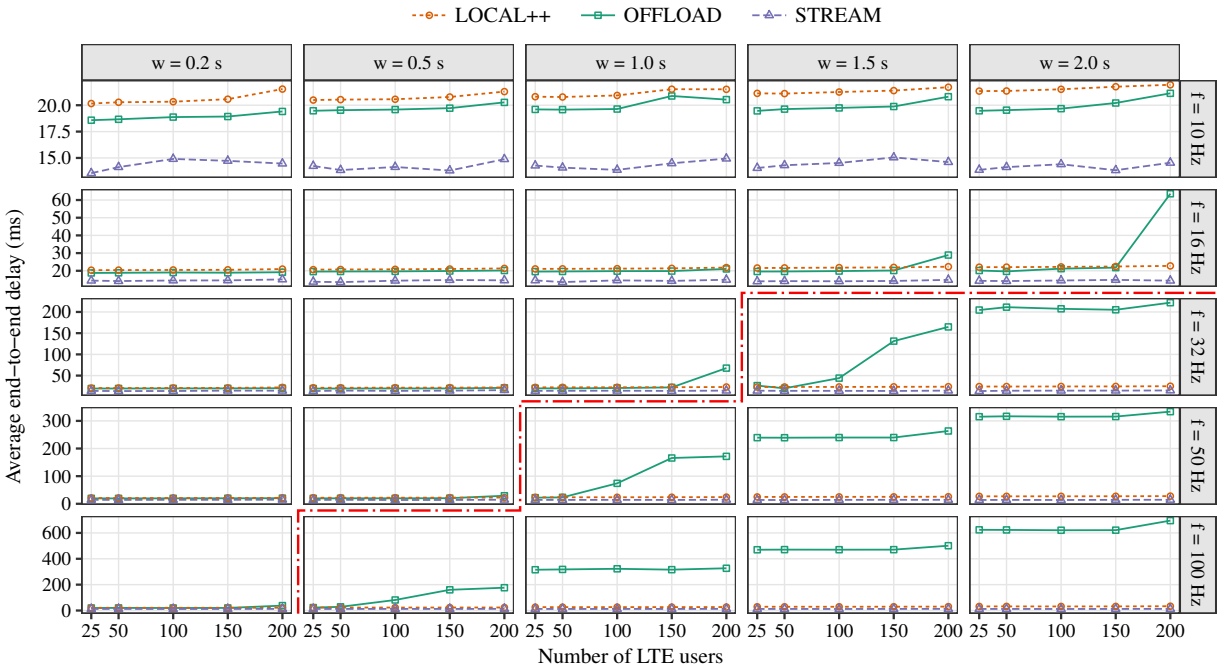



Figure 4: Average end-to-end delay vs. operation scheme, number of LTE users, window length, and sampling frequency.

from about 20–34 ms in all cases. This metric in the STREAM scheme is almost stable with measured values ranging from 12–15 ms.

Looking at the OFFLOAD scheme, we can notice that the latency can vary from about 20–700 ms depending on the values of the user density, window length, and sampling frequency. For $w = 0.2$ s or $f \leq 16$ Hz, this scheme always exposes latency less than 65 ms, regardless of the user density. In the remaining cases, when the product of the window length and the sampling frequency reaches about 50 or more (i.e, the part of Figure 4 below the red line), the end-to-end delay starts to increase really fast as any of these parameters get higher values. For example, with $w = 0.5$ s and $f = 100$ Hz, the delay rises from about 22–176 ms with the user densities increasing from 25–200.

The explanation for these observations mainly lies in the size of messages transmitted in each scheme. While the message sizes used in the LOCAL++ and STREAM are fixed and small, i.e., 300 B and 36 B, respectively, this value grows rapidly relative to the increases of the window length and the sampling frequency in the OFFLOAD scheme, i.e., up to 7200 B for $w = 2.0$ s and $f = 100$ Hz. Moreover, higher density of LTE users mean higher connections to the server, which can cause overload issues, and therefore increase packet latency.

## VI. 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. First, even though the STREAM scheme offers stable and very low end-to-end delay, it is inefficient for smartphones regarding energy consumption. This scheme, therefore, is not recommended for our proposed system, in which limited smartphone battery life is a bottleneck. It is also evident that the OFFLOAD scheme can save more energy on smartphones especially for small windows lengths, while the LOCAL++ scheme produces comparable or lower average end-to-end delays in all cases.

Applying these findings to our proposed Car2P safety system, we can see that the OFFLOAD scheme outperforms the LOCAL++ scheme for short window lengths (e.g., $w = 0.2$ s), or in other words, when smartphones need to quickly update their context information to the server. This situation can be encountered when pedestrians are in high-risk scenarios like reaching or crossing roads, in which the latest information from their smartphones needs to be sent to the server as quickly as possible for collision prediction. Besides, regardless of the window length, the OFFLOAD scheme is still a better option if the smartphone uses low sensor sampling frequencies (e.g., $f \leq 16$ Hz). This circumstance occurs when the pedestrians are in risk-free situations where the classification results are not critical.

In other cases, with long window lengths and high frequency, the LOCAL++ scheme consumes a similar amount of energy or slightly higher than the OFFLOAD scheme, but in return, it shows much better efficiency in terms of message end-to-end delay. These conditions are appropriate for low-risk contexts, in which pedestrian smartphones could reduce the updating rate but still needs to provide accurate information for tracking and prediction processes. This requires a huge number of raw data collected by setting a high sampling frequency. In addition, higher densities of LTE users lead to higher packet latencies. The LOCAL++, therefore, is a better approach for these cases.

## VII. Conclusion

In this paper, we presented an adaptive energy-efficient approach for smartphone context information calculation in our proposed MEC-based Car2P safety system. Our approach considers the possibility of sending raw data collected on smartphones to the MEC server for the computation instead of local execution. In order to evaluate the performance of the suggested approach, first, we implemented a machine learning algorithm for real-time recognizing pedestrian activities as a target application for our investigation. Second, we conducted experimental studies to measure the local processing time as well as the energy consumed by the application in different operation schemes. Finally, the integration of the time measurements on the smartphone and, for the first time, person objects into the Open Source vehicular network simulation framework Veins LTE, allowed us to examine the message end-to-end delay in our proposed safety system. Combining the obtained results from both experimental and simulation studies, we are now able to answer the question of *when* smartphones should offload their context information calculation to the server to be beneficial to the whole safety system. Our results indicate that an offloading scheme is most suitable in case the smartphone is requested to update context information with short periods or to sample sensor data at low frequencies; otherwise, locally performing the calculation is a better option.

## References


[1] *Global status report on road safety 2015*. World Health Organization, 2015.

[2] C. Bila, F. Sivrikaya, M. A. Khan, and S. Albayrak, "Vehicles of the Future: A Survey of Research on Safety Issues," *IEEE Transactions on Intelligent Transportation Systems*, vol. 18, no. 5, pp. 1046–1065, May 2017.

[3] K. David and A. Flach, "CAR-2-X and Pedestrian Safety," *IEEE Vehicular Technology Magazine*, vol. 5, no. 1, pp. 70–76, Mar. 2010.

[4] X. Wu, R. Miucic, S. Yang, S. Al-Stouhi, J. Misener, S. Bai, and W.-h. Chan, "Cars Talk to Phones: A DSRC Based Vehicle-Pedestrian Safety System," in *80th IEEE Vehicular Technology Conference (VTC2014-Fall)*. Vancouver, BC, Canada: IEEE, Sep. 2014.

[5] P. Choi, J. Gao, N. Ramanathan, M. Mao, S. Xu, C.-C. Boon, S. A. Fahmy, and 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)*. La Jolla, CA: ACM, Aug. 2014, pp. 207–212.

[6] S. Chen, J. Hu, Y. Shi, Y. Peng, J. Fang, R. Zhao, and L. Zhao, "Vehicle-to-Everything (V2X) Services Supported by LTE-based Systems and 5G," *IEEE Communications Standards Magazine*, vol. 1, no. 2, pp. 70–76, Jul. 2017.

[7] G. Araniti, C. Campolo, M. Condoluci, A. Iera, and A. Molinaro, "LTE for Vehicular Networking: A Survey," *IEEE Communications Magazine*, vol. 51, no. 5, pp. 148–157, May 2013.

[8] S. Tang, K. Saito, and 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)*. Yokohama, Japan: IEEE, Nov. 2015, pp. 41–47.

[9] A. Rostami, B. Cheng, H. Lu, J. B. Kenney, and 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)*. New York, NY: ACM, Oct. 2016, pp. 22–29.

[10] M. Bagheri, M. Siekkinen, and J. K. Nurminen, "Cloud-Based Pedestrian Road-Safety with Situation-Adaptive Energy-Efficient Communication," *IEEE Intelligent Transportation Systems Magazine*, vol. 8, no. 3, pp. 45–62, 2016.

[11] S. Loewen, F. Klingler, C. Sommer, and 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*. Torino, Italy: IEEE, Nov. 2017, pp. 43–44.

[12] R. B. Zadeh, M. Ghatee, and H. R. Eftekhari, "Three-Phases Smartphone-Based Warning System to Protect Vulnerable Road Users Under Fuzzy Conditions," *IEEE Transactions on Intelligent Transportation Systems*, vol. 19, no. 7, pp. 2086–2098, Jul. 2018.

[13] M. Shoaib, S. Bosch, O. D. Incel, H. Scholten, and P. J. Havinga, "A Survey of Online Activity Recognition Using Mobile Phones," *Sensors*, vol. 15, no. 1, pp. 2059–2085, Jan. 2015.

[14] C. A. Ronao and S.-B. Cho, "Human activity recognition with smartphone sensors using deep learning neural networks," *Expert Systems with Applications*, vol. 59, pp. 235–244, Oct. 2016.

[15] P. Mach and Z. Becvar, "Mobile Edge Computing: A Survey on Architecture and Computation Offloading," *IEEE Communications Surveys & Tutorials*, vol. 19, no. 3, pp. 1628–1656, Mar. 2017.

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

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

[18] J. J. Anaya, P. Merdrignac, O. Shagdar, F. Nashashibi, and J. E. Naranjo, "Vehicle to pedestrian communications for protection of vulnerable road users," in *IEEE Intelligent Vehicles Symposium (IV 2014)*. Dearborn, MI: IEEE, Jun. 2014, pp. 1037–1042.

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

[20] T. Datta, S. Jain, and 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)*. Philadelphia, PA: IEEE, Oct. 2014, pp. 663–667.

[21] A. Tahmasbi-Sarvestani, H. N. Mahjoub, Y. P. Fallah, E. Moradi-Pari, and O. Abuchaar, "Implementation and Evaluation of a Cooperative Vehicle-to-Pedestrian Safety Application," *IEEE Intelligent Transportation Systems Magazine*, vol. 9, no. 4, pp. 62–75, 2017.

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

[23] Q.-H. Nguyen, J. Blobel, and 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)*. Paris, France: IEEE, Aug. 2016.

[24] S. Joerer, M. Segata, B. Bloessl, R. Lo Cigno, C. Sommer, and F. Dressler, "A Vehicular Networking Perspective on Estimating Vehicle Collision Probability at Intersections," *IEEE Transactions on Vehicular Technology*, vol. 63, no. 4, pp. 1802–1812, May 2014.

[25] G. Avino, M. Malinverno, F. Malandrino, C. E. Casetti, C.-F. Chiasserini, G. Nardini, and S. Scarpina, "Poster: A Simulation-based Testbed for Vehicular Collision Detection," in *9th IEEE Vehicular Networking Conference (VNC 2017)*. Torino, Italy: IEEE, Nov. 2017.

[26] C. Sommer, R. German, and F. Dressler, "Bidirectionally Coupled Network and Road Traffic Simulation for Improved IVC Analysis," *IEEE Transactions on Mobile Computing*, vol. 10, no. 1, pp. 3–15, Jan. 2011.

[27] F. Hagenauer, F. Dressler, and C. Sommer, "A Simulator for Heterogeneous Vehicular Networks," in *6th IEEE Vehicular Networking Conference (VNC 2014), Poster Session*. Paderborn, Germany: IEEE, Dec. 2014, pp. 185–186.

[28] G. Bieber, J. Voskamp, and 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. San Diego, CA: Springer, Jul. 2009, pp. 289–296.