An Edge-based Social Distancing Detection Service to mitigate COVID19 propagation

Adlen Ksentini and Bouziane Brik
EURECOM, Sophia Antipolis, France
Email: name.surname@eurecom.fr

***Abstract*—COVID-19 virus has strongly impacted our everyday life. Without the availability of a vaccine or a well-established and efficient treatment, we have to live with it. One way to mitigate the propagation of the virus is to respect social distancing between persons. Indeed, many governments have adopted it as one of the key solutions to reduce the propagation of the Virus. However, it is difficult to enforce social distancing among the population. In this paper, we propose to combine Internet of Things (IoT) and Multi-access Edge Computing (MEC) technologies to build a service that checks and warns people in near real-time, if they are not respecting the social distancing. The proposed service is composed of a client application side installed on the users' smartphone, which periodically sends GPS coordinates to remote servers sitting at the Edge of the network (i.e. at MEC). The remote servers use a local algorithm to detect and warn users that are not respecting the social distancing. The proposed service respects privacy and anonymity, by hiding the user identity, and is capable to warn in near real-time users thanks to the usage of MEC.**

***Index Terms*—IoT, MEC, COVID-19, Low latency, Smart City**

I. Introduction

COVID-19 is a new virus belonging to the CORONA family. So far, more than 10 million people have been affected in the world, and merely half a million have died due to the direct consequences of the virus [1]. While waiting for a vaccine or an efficient treatment, it is mandatory to find solutions for reducing the spread of the Virus. In other words, the world needs to deal with the Virus while waiting for a treatment or a vaccine. In this context, Information Communications Technology (ICT) technologies can play a key role, where combining IoT and recent other technologies, such as Edge Computing, Software Defined Networking (SDN), Network Function Virtualization (NFV) and Machine Learning (ML), can allow the definition of new applications or services to be used by doctors, government and persons to fight against the spread of the Virus. So far, several use-cases have been envisioned to use IoT as a way to help facing the Virus. For instance, we can mention the case of elderly people, which would be followed up closely. In this case, panic buttons, sensors that monitor movement, energy, and even doors, can be used in hospitals and rest homes, but also for people who still live at home; allowing report anomalies and timely action can be taken. Another use-case can be the monitoring of people in quarantine. Sensors can be used to track people who should live in (self) quarantine, and not always follow the rules and sometimes dare to go outside their zone, posing a high risk of spreading the virus. In addition, new applications for mobile devices have appeared, such as StopCovid application [2], which allows warning persons who have been in contact with infected persons, and hence to track clusters of infected persons and isolate them. The StopCovid application uses Bluetooth in order to save the ID of persons who were close during a certain period. If one of these persons declares in the application that he has been affected, an alarm is sent to all the users who have been in contact with him. The weakness of this application is the usage of Bluetooth (it must be turned on in the smartphone), which is known for its concern with security.

On the other hand, it is well established that one of the key solutions that needs to be adopted to reduce the spreading of the Virus is to keep a minimum distance between persons, namely social distancing. In fact, many studies have shown that there is a minimum distance to keep between persons in order to avoid contagion; the minimum distance varies from 1 meter to 2 meters according to the countries. Therefore, there is a need to have an application or a service that, in run-time, detects if users, located in a certain area, do not respect the social distancing. StopCovid does not include such features as it just records the persons which were in contact, and if one of them is contaminated by the Virus, it has to declare it in the application; then, all the persons who were in contact with that person are warned. Besides being reactive (i.e. after contamination), the StopCovid application needs people to actively participate in the application usage. Other solutions have been developed toward checking, in run-time, the respect of social distancing. We can mention applications using Cameras and ML techniques, such as the system used by Amazon to enforce social distancing at its warehouse [3], or the one used in [4] to track social distancing. However, these applications do not warn concerned persons, but police or managers.

In this work, we propose a novel social distancing detection service that runs at the edge of the network, and aims at detecting if users are not respecting the minimum recommended distances to avoid contamination, and warn them accordingly. To this aim, the proposed service is composed of an application that runs on user's smartphones, equipped with sensors - most notably global navigation satellite system (GNSS), which periodically sends GPS coordinates to a remote application, sitting at the edge. The remote application's role is to compute the distances (Euclidean), using the GPS

coordinates, between users located under the edge server coverage (i.e. geographical location). Users that are close to each other, hence not respecting the social distancing, are warned through messages sent by the remote application to the smartphone. It is worth noting that the proposed service requires low latency communications as users need to be warned rapidly (in near real-time) in case of non-respect of social distancing. Thanks to edge computing, and particularly to the ETSI Multi-access Edge Computing (MEC) system, the proposed service will be deployed at the edge and benefit from the MEC ETSI ecosystem [5]. Besides ensuring low latency communication and hence on near real-time reaction, the usage of MEC will guarantee system scalability, as MEC servers are deployed in a distributed fashion inside a mobile network, hence accommodating a high number of users compared to a centralized solution. Also, the proposed service will ensure privacy and anonymity as no personal information needs to be disclosed in order to use the application. User identifiers are generated and linked only to a valid email address, hence ensuring anonymity. Last but not least, the proposed solution does not need a high involvement from users as in StopCovid. The only requirement is that the application is running on the smartphones, and the end-users have to be alert to the notifications.

The remainder of this paper is organized as follows. Section II gives the envisioned use-case and architecture. Section III introduces our proposed solution and algorithm. Potential applications and extensions of our solution are highlighted in section IV. We discuss the obtained results in section V and conclude the paper in section VI.

II. Envisioned use-case and architecture

Before describing the envisioned architecture, we will start by introducing MEC. The latter is a new trend that enables a new generation of services that operate close to end-users aiming at reducing the end-to-end latency. MEC allows the deployment of two types of service: (i) applications that require low latency access to user plane traffic; (ii) context-aware applications that adapt the delivered service according to users' environment. MEC is an operator-oriented architecture, which adds computing capability in the vicinity of base stations, and proposes an orchestration and management framework to handle the Life Cycle Management (LCM) of edge applications. ETSI is providing specifications to MEC, via the ISG MEC group [6] [7] that released several documents to describe: envisioned use-cases, a reference MEC architecture, a MEC application model (descriptor), MEC services, MEC Orchestrator, etc. Besides running applications at the edge, MEC provides services, accessible via a high-level API, which give information on the mobile users and the cellular base stations context, such as radio channel quality of users; allowing building context-aware applications.

In this work, we envision the system architecture as depicted in Fig. 1, which is composed by MEC servers, users connected to the servers via the application installed on their smartphones. We assume that the network operator uses a set of MEC servers to cover different areas, which allow deploying several instances of the service to guarantee scalability. The size of the area to be covered by a MEC server depends on the density of users. We assume that a MEC server is associated with a set of base stations (or eNB in LTE and geNB in 5G). The number of base stations associated with a MEC server depends on the density of the users. In rural areas, we can imagine having one MEC server covering a high number of macrocells, whereas in dense and urban areas, one MEC server covers a low number of macrocells or a high number of small cells. This deployment can also be envisioned in the context of smart cities, where servers are deployed to cover neighborhoods, and the service is managed by the city. In this case, the service is deployed by the network operator as a Network Slice [8], and fully managed by the city, considered then as the vertical. The social distancing detection application runs inside a container, and can be duplicated on all the MEC servers. Since users are mobile, they can be migrated from one MEC server to another MEC server when they move between two cells, which do not belong to the same MEC server coverage. Readers may refer to [9] for more details on service migration in MEC.

We assume a scenario represented in Fig. 1, where users are spread over locations covered by different MEC servers. The users have installed the client-side of the service on their smartphone, and are already connected with the server-side of the service located at the closest MEC server (i.e. the one that is covering the geographical location of the user). While walking, the client-side of the application periodically communicates the GPS coordinates to the remote detection application server. The latter receives all the GPS coordinates of users under the coverage of the MEC server. A local algorithm is used, which takes as input the GPS coordinates of all users, and gives as output the Id of users that are not respecting the social distancing. The concerned users will receive a warning message from the server, visible directly on their smartphone as a notification; indicating that they are not respecting the social distancing.

On the other hand, the social distancing detection instances are connected to a remote application located in the central cloud, which back-ups the information on how many people have been warned, and the different locations of the persons. It should be noted that the identity of users is protected, only their Ids are disclosed. The cloud application can use the obtained information to report statistics to the government and the city mayor, on the locations where high warning messages have been triggered, which may help to understand and organize for instance the concerned locations (i.e. streets) in order to reduce the warning alarms in the future.

III. Proposed solution and algorithm

As stated earlier, the proposed social distancing detection service is composed of a client application and a server application. The client-side of the application is very simple, it consists in being connected to the remote server (at the edge) and periodically sends the GPS coordinates. The most intelligent part of the service is at the server-side, which uses

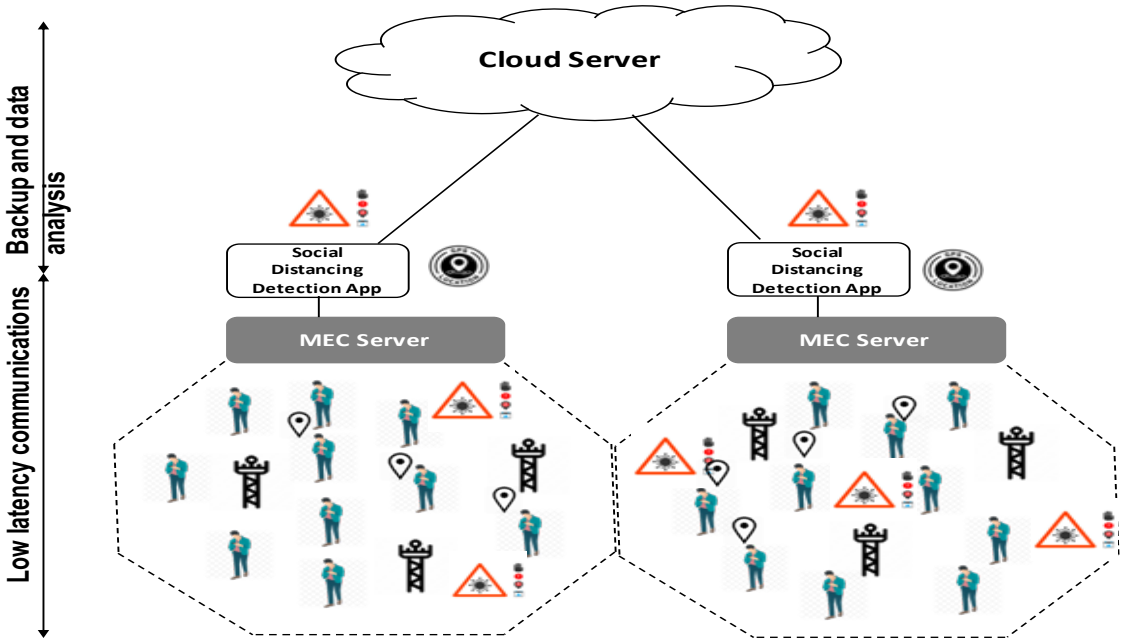

Fig. 1. MEC-Based Architecture for Social distancing detection.

the received GPS coordinates of users in order to calculate the distances and generates warning messages if deemed appropriate. The proposed algorithm to be run at the server-side is shown in Algorithm 1. We distinguish three steps, as depicted in Fig. 2: (i) The collection of all the GPS coordinates of users connected to the server (i.e. under its coverage); (ii) The computation of the distance between users, in a pair by pair based; (iii) The detection of users not respecting the distance threshold, and the warning message generation. For step 1, the collected GPS coordinates are done periodically, and saved in two vectors: $\Phi_t(i)$ corresponding to the latitude coordinates and $\Lambda_t(i)$ for the longitude coordinates, indexed by the user id (i.e. i). In step 2, a function $Distance$ is called for each pair of users (i, j) and takes as inputs $\Phi_t(i)$, $\Phi_t(j)$, $\Lambda_t(i)$, and $\Lambda_t(j)$. The details of this function are not included in this work, but it can be based on any well-known method such as: Pythagore or Sinus law. Then, the distance function saves the distances in a matrix $dist(i, j)$ corresponding to the distance between i and j. In step 3, the algorithm checks for each user the distance with the other users by pair. If the distance is not respected, the concerned user (noted i) is warned, and the loop is stopped for that user, i.e. no need to check for other users as he is already close to at least one person, and should be warned. User (j) is also warned, if he has not been warned before; otherwise, he is ignored as no need to warn him twice. The two last actions allow reducing the execution time of the algorithm, as the loop is stopped when a person is not respecting a distance with at least one other person. In fact, in terms of computation complexity, the proposed algorithm has a complexity of $N \times M$, where $1 \leq M \leq N - 1$, as the first loop stops when the first person non respecting the social distancing is found. It is worth recalling that N is the number of users connected to the MEC server (i.e. users under the coverage). This number is kept low thanks to the distributed MEC architecture that allows duplicating the instances of the server-side of the application.

As mentioned before, the algorithm will run periodically. To avoid synchronizing all the clients with the servers, we propose that each server opens a window or a period to collect the GPS coordinates of users sent by the client applications. At the end of the period, the algorithm is run and concerned persons are warned. The duration of the period is critical and needs to be well investigated and tuned when deploying the service. Indeed, long period duration may decrease the accuracy of the algorithm to detect users that are not respecting the social distancing, but reduce the exchanges of messages with the client-side of the application. One way to improve the algorithm in order to be less dependant from this period is to use, in addition to the GPS coordinate, the person speed and the acceleration that can be obtained from the smartphone's accelerometer. However, this will increase the complexity of the algorithm, and hence the time to run it. The current version of the algorithm is very fast to run. Even if we consider a short period to improve accuracy, the size of the message to exchange is negligible, so no impact on the network is expected. Indeed, the messages sent by the clients contain four fields: User Id, timestamp, GPS coordinates (longitude and latitude), which require only a few bytes of data.

IV. Discussion and Potential Applications

A. Privacy and anonymity

In order to keep users privacy, the proposed social distancing detection service does not require personal details, such as name and mail address to run. At the registration step, a

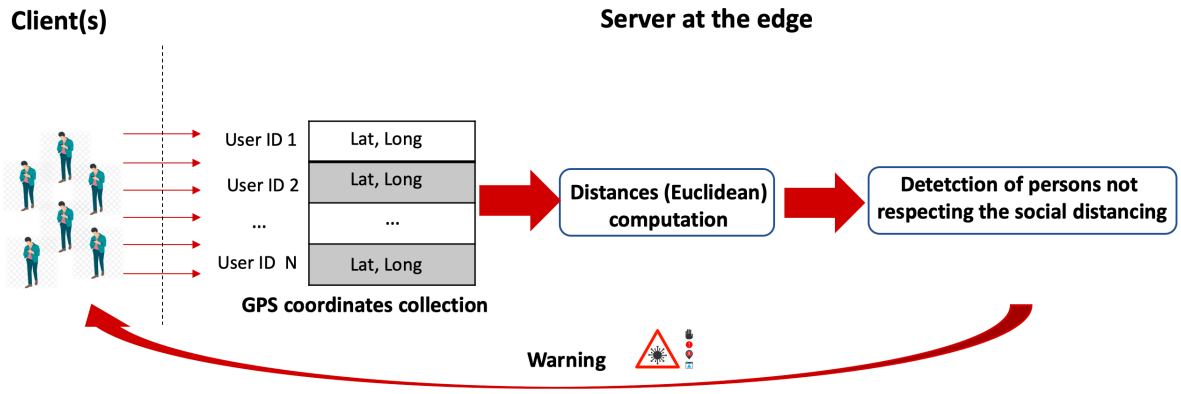

Fig. 2. The concept of the social distancing detection service

Algorithm 1 Distance Detection

Require: GPS coordinates Φ_t, Λ_t, $threshold$.
Ensure: Send Warning Messages to Users.

for i from 1 to $N-1$ **do**
  for j from $i+1$ to N **do**
    Dist (i,j) = Distance (i, j)
  end for
end for
while $i \leq N$ and $Warn(i) == False$ **do**
  for j from 1 to N **do**
    if $Dist(i,j) \geq$ threshold **then**
      Send a warning Message to user i
      if $Warn(j) == False$ **then**
        Send a warning Message to user j
      end if
    end if
  end for
end while

user may create an account just by providing a valid email address (any email valid email address no link with the identity of the user). Then, the system generates an account ID linked to the email address and a password is requested and associated with the user account. Everything is stored in a distributed Data Base (DB) shared by all the instances of the application (server-side), which helps to handle migration of users among the MEC servers. When the user installs the application on its smartphone, he needs to have access to the GNSS/accelerometer of the smartphone and to use the login (account ID) and its associated password. The client part of the application then opens a socket with the remote server, and after being authenticated, the communication flows starts between the Client and Server. At the server-side, the only information which is available is the account ID and its associated email, so no direct link can be established with the person's identity.

B. Data Collection and Analytic

As mentioned earlier, all the server instances report statistics to a cloud back-end server; for example, on the number of warnings sent during the days, the GPS coordinates (mobility of user) based only on the ID of users to keep anonymity. This information is critical in order to fight against the virus propagation. One extension of the proposed service is to use the collected data to help the local government or city major, for instance, to understand the correlation between the number of generated alerts and the location. This will allow detecting the locations where the social distancing is not well respected, and then actions can be taken to better organize the concerned locations.

Another direction we are thinking about is the usage of the warning alerts to detect contamination and the size of the cluster (i.e. a group of contaminated persons that are linked together). Indeed, ML techniques can be used, which take as input the alarms and the locations as well as inputs from the health agency about the locations of contaminated persons. The ML tool will be trained to find a relation between the alarms, the locations and the number of contaminated persons. The ML will be later used to predict according to the generated alarms the probability of contaminated people in the considered location, hence helping to detect a cluster of contaminated people, and take actions such as quarantine or lock down those locations. Moreover, if the probability to be infected in a location is high, the warning that is sent to users, which are not respecting social distancing, may include this probability as additional information. This can be an incentive to respect the social distancing, or to leave that area.

On the other hand, the proposed service can behave as the StopCovid application, by allowing a user to indicate through the application that it has been contaminated. Then the algorithm will be run at the central cloud, by taking as input only that user ID, and applied on the collected data, for a window of one week. All users that were not respecting the social distancing with that contaminated person will be warned and invited to go as soon as possible to check if they

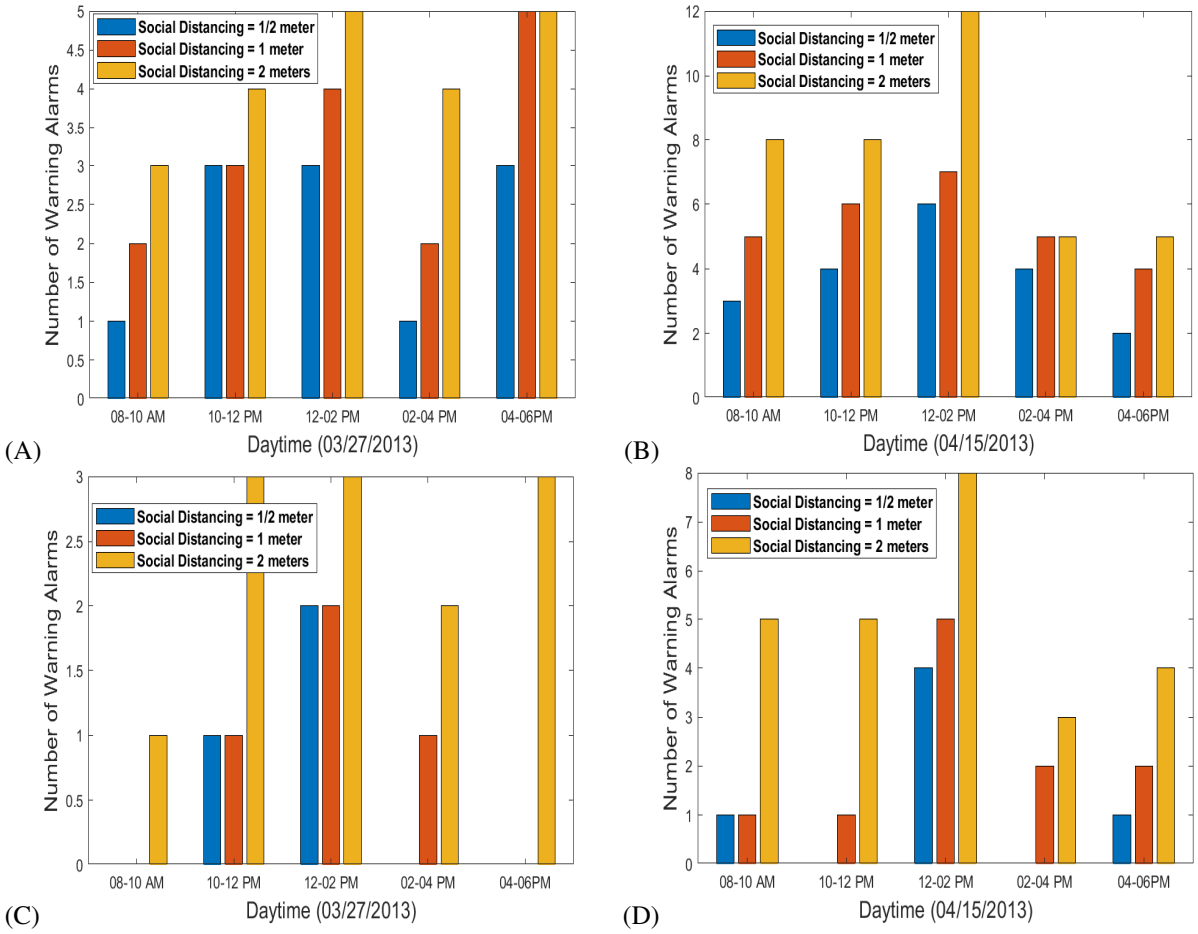

Fig. 3. The number of generated alarms in two different days with different GPS Collection Frequency (CF). (A) and (B) CF each 100 ms. (C) and (D) CF each 1s.

were contaminated or not.
Finally, the gathered GPS coordinates can be used in real-time to identify persons sharing the same mobility behavior in terms of daily itinerary, working in the same company, living in the same district, etc. Therefore, the ML discussed in the preceding paragraph can be extended to associate the probability of contamination according to the group of users.

V. Performance Evaluation

We evaluate the proposed social distancing detection service through two different methods. First, we implement our distance detection scheme in Python. As input for the algorithm, we used a real existing dataset of mobile people. Second, we have implemented a prototype of the application and tested it using the EURECOM's MEC platform [10], developed on top of OpenAirInterface (OAI) [11]. Whilst the first method allows us to check the efficiency of the social distancing detection algorithm, introduced in Algorithm 1, the second method permits to show how MEC can help to ensure the near real-time communication, and hence guarantee short-latency to receive warnings when non respecting the distances.

A. StudentLife Dataset

In the first method of evaluation, we have used a real dataset named *StudentLife* [12] to apply our social distancing detection algorithm on it, and extract its performances. *StudentLife* is a large and a longitudinal dataset that contains sensing data from the phones of a class of 48 students over a 10 week spring term. The StudentLife dataset includes rich and in-depth information over 53 GB of sensed data. Among these data, it comprises location-based data, corresponding to real-time GPS coordinates obtained from students' smartphones. It is worth noting that this dataset is anonymized in order to protect the privacy of the participant students. The dataset is used to provide GPS coordinates to our algorithm, which in turn will detect users that are not respecting the social distancing and hence generate alerts accordingly. We used this dataset instead of simulated GPS coordinate to see the behaviour of our algorithm when facing a real mobility model.

B. Evaluation of Distance Detection Scheme

Fig. 3 depicts the number of warning messages sent to the students during two different days, while varying the social distancing that students have to respect as well as

the Collection Frequency (CF) (or period to collect the data and run the algorithm) of persons' GPS coordinates (each 100ms for Fig. 3 (A) and (B), and each 1s for Fig. 3 (C) and (D)). It should be noted that in these results, we select randomly five students studying in the same class and we focus on two different days ($03/27/2013$ and $04/15/2013$); we focused only on 5 persons. Clearly, we observe that the number of sent warning alarms increases, as the social distancing threshold increases. We argue this by the fact that initially the students are far away from each other (more than half a meter); but if we increase the social distancing threshold, the number of generated alarms increases as well. We also remark that the number of warning alarms decreases as we decrease the CF of GPS coordinates (cf. Fig. 3 (C) and (D)). In this case, when decreasing the CF, the MEC server collects less fresh GPS coordinates, which reduces the accuracy of the algorithm. Indeed, the non-respect of social distancing between two successive messages (i.e. GPS coordinates) will not be detected, leading to generate less number of warning messages. In this context, it is preferable to increase the CF aiming at increasing the application accuracy to detect persons that are not respecting the social distancing. Although this solution means increasing the exchanged messages between the client and server, the burden on the network remains low. Only a few bytes of data is needed to send one message, which is very negligible compared to the expected high data rate in the new generation of mobile networks like 5G. One solution to keep the algorithm accuracy while reducing CF is to use additional inputs, such as users' speed and acceleration in order to predict future user positions; which in turn adds complexity to the distance detection algorithm.
On the other hand, for each social distancing threshold, we see that from 12pm to 02pm the number of alarms is higher than in the other time periods. This is mainly due to the fact that in this time period, between 12pm to 02pm, students are out of their classes (and/or school), which results in more non-respect of the social distancing, as the distances between student are becoming shorter.

C. Prototype

We have developed a prototype of the service, where the client part runs on top of an Android phone, and the server in a docker container at a MEC server. For the infrastructure we have used the OAI platform to provide 4G connectivity, and the MEC platform developed by EURECOM to run the application server at the edge. We have done the test with two smartphones connected to the server. We have measured the latency at the client-side when a notification is obtained. The measures have been done at the application level, which runs on top of a TCP connection. Here, since only two phones have been tested, the latency due to algorithm 1 execution is negligible; the shown values include mainly the network latency.

Table I illustrates the obtained results, showing: the average, the maximum, the minimum, and the standard deviation of the measured latency. We see that using a MEC server allows reducing the end-to-end latency to an average of 14.63ms, which means that the user is warned practically in near real-time if the social distancing is not respected, which is one of the ultimate objectives of the proposed service.

TABLE I
MEASURED END-TO-END LATENCY

Average	Maximum	Minimum	Deviation
14.63 ms	55 ms	11 ms	3.59 ms

VI. CONCLUSION

In this paper, we introduced a novel social distancing detection service that runs at the edge of the network. It aims at detecting, in near real-time, if persons are not respecting the minimum recommended distances to avoid contamination, and hence to reduce the propagation of COVID-19. Our scheme relies on the scalable MEC ETSI system, which ensures low latency communication and hence guarantee a near-real time reaction. In addition, the proposed scheme ensures users' privacy since no personal information is required to run the service.
To demonstrate the feasibility of our scheme, we evaluated its behaviour using a real dataset of mobile people as well as a prototype based on the OAI platform. The obtained results showed the efficiency of our scheme in alerting users in near real-time regarding the minimum distance to respect, regardless of users' mobility behavior (in indoor or outdoor). We expect, in future work, to improve the proposed algorithm by considering other parameters, such as speed and acceleration, and use ML to predict the locations where the probability of contamination is high. In addition, we aim to test our solution in a real deployment, with a high number of users.

REFERENCES

[1] "Coronavirus disease (covid-19)," https://www.who.int/emergencies/diseases/novel-coronavirus-2019, accessed: 2020-07-14.
[2] "Stopcovid," https://www.economie.gouv.fr/stopcovid, accessed: 2020-07-14.
[3] "Amazon using cameras to enforce social distancing rules at warehouse," https://www.cnbc.com/2020/06/16/amazon-using-cameras-to-enforce-social-distancing-rules-at-warehouses.html, accessed: 2020-07-14.
[4] "Artificially intelligent (ai) cameras track social distancing," https://www.mygreatlearning.com/blog/artificially-intelligent-ai-cameras-track-social-distancing/, accessed: 2020-07-14.
[5] A. Huang, N. Nikaein, T. Stenbock, A. Ksentini, and C. Bonnet, "Low latency MEC framework for sdn-based LTE/LTE-A networks," in *IEEE International Conference on Communications, ICC 2017, Paris, France, May 21-25, 2017*. IEEE, 2017, pp. 1–6.
[6] *Mobile Edge Computing (MEC); Mobile Edge Management; Part 2: Application lifecycle, rules and requirements management*, ETSI Group Specification MEC 010, July 2017.
[7] *Mobile Edge Computing (MEC); Framework and Reference Architecture*, ETSI Group Specification MEC 003, March 2016.
[8] A. Ksentini and P. Frangoudis, "Toward slicing-enabled multi-access edge computing in 5g," *IEEE Network*, vol. 34, no. 1, pp. 99–105, January 2020.
[9] P. A. Frangoudis and A. Ksentini, "Service migration versus service replication in multi-access edge computing," in *2018 14th International Wireless Communications Mobile Computing Conference (IWCMC)*, 2018, pp. 124–129.

[10] S. Arora, P. Frangoudis, and A. Ksentini, "Exposing radio network information in a mec-in-nfv environment: the rnisaas concept," in *Proceedings of the 2019 IEEE Network Softwarization Conference*, ser. Netsoft'19, 2019.
[11] N. Nikaein, M. K. Marina, S. Manickam, A. Dawson, R. Knopp, and C. Bonnet, "Openairinterface: A flexible platform for 5g research," *SIGCOMM Comput. Commun. Rev.*, vol. 44, no. 5, p. 33–38, Oct. 2014.
[12] R. Wang, F. Chen, Z. Chen, T. Li, G. Harari, S. Tignor, X. Zhou, D. Ben-Zeev, and A. T. Campbell, "Studentlife: Assessing mental health, academic performance and behavioral trends of college students using smartphones," in *Proceedings of the 2014 ACM International Joint Conference on Pervasive and Ubiquitous Computing*, ser. UbiComp '14, New York, NY, USA, 2014, p. 3–14.

BIOGRAPHIES

ADLEN KSENTINI is an IEEE COMSOC distinguished lecturer. He obtained his Ph.D. degree in computer science from the University of Cergy-Pontoise in 2005. Since March 2016, he is a professor in the Communication Systems Department of EURECOM. He has been working on several EU projects on 5G, Network Slicing, and IoT.

BOUZIANE BRIK received his Ph.D degree from Laghouat and La Rochelle (France) universities in 2017. He is currently a Research Fellow at the Communication Systems Department, EURECOM, France. He has been working on network slicing in the context of H2020 European project on 5G. His research interests include also Internet of Things (IoT), IoT in industrial systems, Smart grid, and vehicular networks.