An Edge-Based Social Distancing Detection Service to Mitigate COVID-19 Propagation

Adlen Ksentini and Bouziane Brik

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 article, we propose to combine 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 practicing 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 who are not practicing social distancing. The proposed service respects privacy and anonymity by hiding the user identity, and is capable of warning users in near real time thanks to the usage of MEC.

Introduction

COVID-19 is a new virus belonging to the coronavirus family. So far, more than 10 million people have been affected in the world, and over 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) can play a key role, where combining IoT and other recent 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, governments, 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 in facing the virus. For instance, we can mention the case of elderly people, which would be followed up closely. In this case, panic buttons, and 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 anomaly reporting so that 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, but do 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 the StopCovid application [2], which allows warning persons who have been in contact with infected persons, and hence 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 they have been affected, an alarm is sent to all the users who have been in contact with them. 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 spread 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 m to 2 m according to the country. Therefore, there is a need to have an application or a service that, in runtime, detects if users, located in a certain area, do not respect social distancing. StopCovid does not include such features as it just records the persons who 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 runtime, the respect of social distancing. We can mention applications using cameras and machine learning (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 to detect 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, a 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, and hence not respecting 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 European Telecommunications Standards Institute (ETSI) multi-access edge computing (MEC) system, the proposed service will be deployed at the edge and benefit from the ETSI MEC ecosystem [5]. Besides ensuring low-latency communication and hence 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 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 article is organized as follows. We give the envisioned use case and architecture. We introduce

Digital Object Identifier: 10.1109/IOTM.0001.2000138

our proposed solution and algorithm. Potential applications and extensions of our solution are highlighted. We discuss the obtained results and conclude the article.

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 which operate close to end users aimed at reducing the end-to-end latency. MEC allows the deployment of two types of service:

- Applications that require low-latency access to user plane traffic
- Context-aware applications that adapt the delivered service according to users' environments

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 on MEC via the ISG MEC group [6, 7], who released several documents to describe: envisioned use cases, a reference MEC architecture, a MEC application model (descriptor), MEC services, MEC orchestrator, and so on. Besides running applications at the edge, MEC provides services, accessible via a high-level application programming interface (API), which give information in the mobile users' and cellular base stations' contexts, 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 of MEC servers, and 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 allows 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 eNBs in LTE and geNBs 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 that do not belong to the same MEC server coverage. Readers may refer to [9] for more details on service migration in MEC.

We assume the 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 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 social distancing.

On the other hand, the social distancing detection instances are connected to a remote application located in the central cloud, which backs up 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 locations where high warning messages have been triggered, which may help, for instance, to understand and organize the concerned locations (i.e., streets) in order to reduce warning alarms in the future.

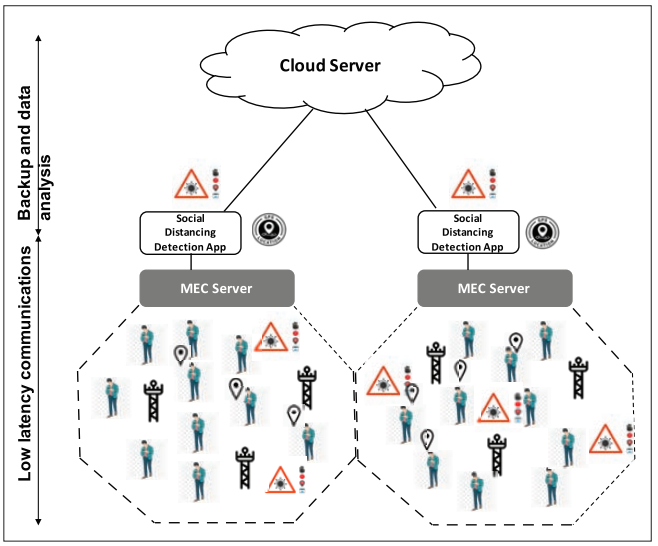

FIGURE 1. MEC-based architecture for social distancing detection.

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, consisting of being connected to the remote server (at the edge) and periodically sending GPS coordinates. The most intelligent part of the service is at the server side, which uses the received GPS coordinates of users in order to calculate the distances and generate warning messages if deemed appropriate.

The proposed algorithm to be run at the server side is shown as Algorithm 1. We distinguish three steps, as depicted in Fig. 2:

1. The collection of all the GPS coordinates of users connected to the server (i.e., under its coverage)
2. The computation of the distance between users, on a pair-by-pair basis
3. The detection of users not respecting the distance threshold, and the warning message generation

For step 1, the collection of GPS coordinates is 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 from 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 she 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 there is 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 $N \times M$,

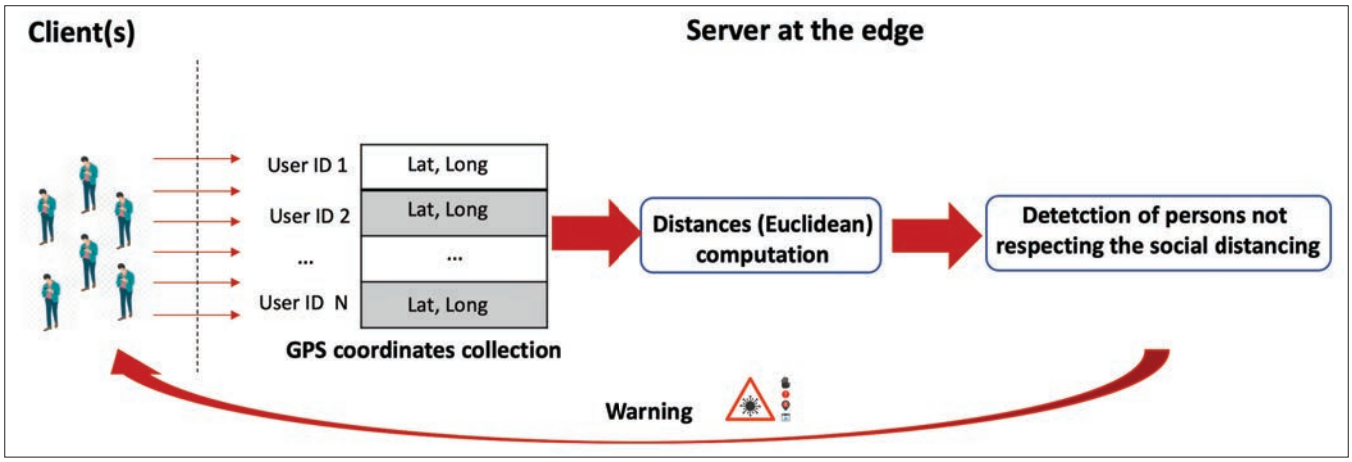

FIGURE 2. The concept of the social distancing detection service.

where $1 \leq M \leq N - 1$, as the first loop stops when the first person not respecting 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, which 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 who are not respecting 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 dependent of this period is to use, in addition to the GPS coordinate, persons' 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.

Discussion and Potential Applications

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 user may create an account just by providing a valid email address (any valid email address; no link to 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 database (DB) shared by all the instances of the application (server side), which helps to handle migration of users among MEC servers. When the user installs the application on her smartphone, she 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 that is available is the account ID and its associated email, so no direct link can be established with the person's identity.

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 users) 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 mayor, for instance, to understand the correlation between the number of generated alerts and the location. This will allow detecting locations where social distancing is not well respected; then actions can be taken to better organize the locations concerned.

Another direction we are thinking about is the usage of warning alerts to detect contamination and the size of the cluster (i.e., a group of contaminated persons who 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 used later 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 locking down those locations. Moreover, if the probability of being infected in a location is high, the warning sent to users who are not respecting social distancing may include this probability as additional information. This can be an incentive to respect social distancing or to leave that area.

On the other hand, the proposed service can behave like the StopCovid application, by allowing a user to indicate through the application that he 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 who were not respecting the social distancing with that contaminated person will be warned and invited to go as soon as possible to test for contamination.

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, and so on. Therefore, the ML discussed in the preceding paragraph can be extended to associate the probability of contamination according to the group of users.

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 EURECOM's MEC platform [10], developed on top of OpenAirInterface (OAI) [11]. While the first method allows us to check the efficiency of the social distancing detection algorithm, introduced in Algorithm 1, the second method permits us to show how MEC can help to ensure near-real-time communication, and hence guarantee short latency to receive warnings when not respecting the distances.

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 performance. *StudentLife* is a large and 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 who are not respecting the social distancing and hence generate alerts accordingly. We used this dataset instead of simulated GPS coordinates to see the behavior of our algorithm when facing a real mobility model.

Evaluation of the Distance Detection Scheme

Figure 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 100 ms for Figs. 2a and 2b, and each 1 s for Figs. 2c and 2d). It should be noted that in these results, we randomly select 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 five 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 (Figs. 2c, 2d). 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 generation of fewer warning messages. In this context, it is preferable to increase the CF, aiming to increase the application accuracy to detect persons who are not respecting social distancing. Although this solution means increasing the exchanged messages between the client and the server, the burden on the network remains low. Only a few bytes of data are 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 12 p.m. to 02 p.m., the number of alarms is higher than in the other time periods. This is mainly due to the fact that in this time period, students are out of their classes (and/or school), which results in more non-respect of social distancing, as the distances between students become shorter.

Prototype

We have developed a prototype of the service, where the client part runs on top of an Android phone, and the server is 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 measurements 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 mainly include the network latency.

Require: GPS coordinates Φ_t, Λ_t, *threshold*.
Ensure: Send Warning Messages to Users.
1:
2: **for** i from 1 to $N-1$ **do**
3: **for** j from i+1 to N **do**
4: Dist (i, j) = Distance (i, j)
5: **end for**
6: **end for**
7: **while** $i \leq N$ and *Warn* $(i) ==$ False **do**
8: **for** j from 1 to N **do**
9: **if** *Dist* $(i, j) \geq$ threshold **then**
10: Send a warning Message to user i
11: **if** *Warn*$(j) ==$ False then
12: Send a warning Message to user j
13: **end if**
14: **end if**
15: **end for**
16: **end while**

ALGORITHM 1. Distance detection.

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

TABLE 1. Measured end-to-end latency.

Table 1 illustrates the obtained results, showing the average, maximum, minimum, and standard deviation of the measured latency. We see that using a MEC server allows reducing the end-to-end latency to an average of 14.63 ms, which means that the user is warned practically in near real time if social distancing is not respected, which is one of the ultimate objectives of the proposed service.

Conclusion

In this article, we introduce a novel social distancing detection service that runs at the edge of the network. It aims to detect, 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 behavior using a real dataset of mobile people as well as a prototype based on the OAI platform. The obtained results show the efficiency of our scheme in alerting users in near real time regarding the minimum distance to respect, regardless of users' mobility behavior (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] WHO, "Coronavirus Disease (covid-19)"; https://www.who.int/emergencies/diseases/novel-coronavirus-2019, accessed July 14, 2020.
[2] "Stopcovid"; https://www.economie.gouv.fr/stopcovid, accessed July 14, 2020.

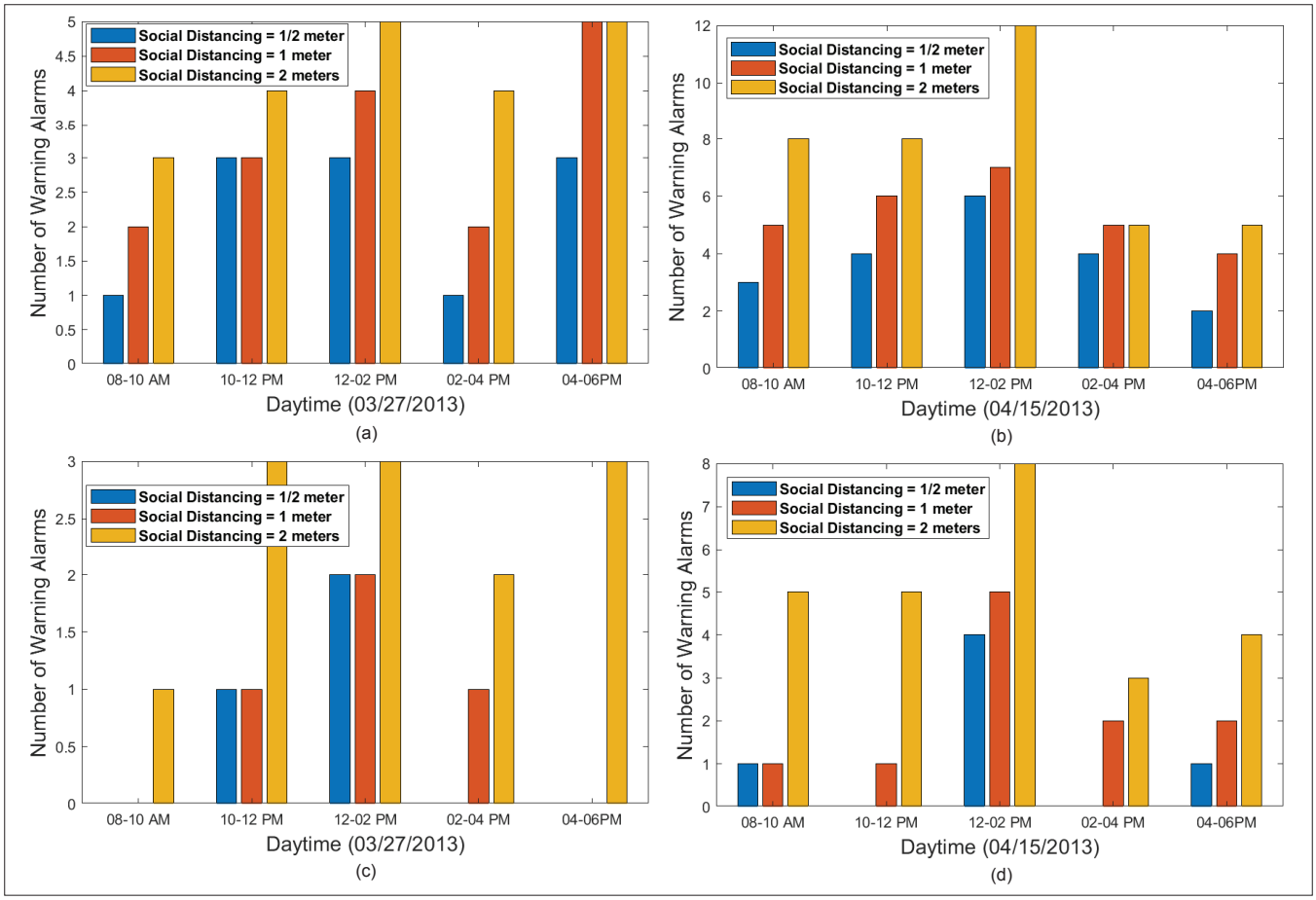

FIGURE 3. The number of generated alarms in two different days with different GPS collection frequency (CF): a), b) CF each 100 ms.; c), d) CF each 1 s.

[3] "Amazon Using Cameras to Enforce Social Distancing Rules at Warehouse"; https://www.cnbc.com/2020/06/16/amazon-using-camerasto-enforce-social-distancing-rules-at-warehouses.html, accessed July 14, 2020.

[4] "Artificially Intelligent (AI) Cameras Track Social Distancing"; https://www.mygreatlearning.com/blog/artificially-intelligent-aicameras-track-social-distancing/, accessed July 14, 2020.

[5] A. Huang *et al.*, "Low Latency MEC Framework for SDN-Based LTE/LTE-A Networks," *IEEE ICC 2017*, Paris, France, May 21–25, 2017, pp. 1–6.

[6] "Mobile Edge Computing (MEC); Mobile Edge Management; Part 2: Application Life Cycle, Rules and Requirements Management," ETSI Group Spec. MEC 010, July 2017.

[7] "Mobile Edge Computing (MEC); Framework and Reference Architecture," ETSI Group Spec. MEC 003, Mar. 2016.

[8] A. Ksentini and P. Frangoudis, "Toward Slicing-Enabled Multi-Access Edge Computing in 5G," *IEEE Network*, vol. 34, no. 1, Jan./Feb. 2020, pp. 99–105.

[9] P. A. Frangoudis and A. Ksentini, "Service Migration Versus Service Replication in Multi-Access Edge Computing," *2018 14th Int'l. Wireless Commun. Mobile Computing Conf.*, 2018, pp. 124–29.

[10] S. Arora, P. Frangoudis, and A. Ksentini, "Exposing Radio Network Information in a Mec-in-Nfv Environment: The Rnisaas Concept," *Proc. 2019 IEEE Network Softwarization Conf.*, ser. Netsoft'19, 2019.

[11] N. Nikaein *et al.*, "Openairinterface: A Flexible Platform for 5G Research," *SIGCOMM Comp. Commun. Rev.*, vol. 44, no. 5, Oct. 2014, p. 33–38.

[12] R. Wang *et al.*, "Studentlife: Assessing Mental Health, Academic Performance and Behavioral Trends of College Students Using Smartphones," *Proc. 2014 ACM Int'l. Joint Conf. Pervasive and Ubiquitous Computing*, ser. UbiComp '14, New York, NY, USA, 2014, p. 3–14.

Biographies

Adlen Ksentini (adlen.ksentini@eurecom.fr) is an IEEE Communications Society Distinguished Lecturer. He obtained his Ph.D. degree in computer science from the University of Cergy-Pontoise in 2005. Since March 2016, he has been 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 (bouziane.brik@eurecom.fr) received his Ph.D. degree from Laghouat and La Rochelle Universities, France, 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 the H2020 European project on 5G. His research interests also include IoT, IoT in industrial systems, smart grid, and vehicular networks.