A Context, Location and Preference-Aware System for Safe Pedestrian Mobility

Constantinos Costa
Dept. of Computer Science
University of Pittsburgh
costa.c@cs.pitt.edu

Brian T. Nixon
Dept. of Computer Science
University of Pittsburgh
nixon.b@cs.pitt.edu

Sayantani Bhattacharjee
Dept. of Computer Science
University of Pittsburgh
sab301@pitt.edu

Benjamin Graybill
Dept. of Computer Science
University of Pittsburgh
bcg36@pitt.edu

Demetrios Zeinalipour-Yazti
Dept. of Computer Science
University of Cyprus
dzeina@cs.ucy.ac.cy

Walter Schneider
Learning Research & Development Center
University of Pittsburgh
wws@pitt.edu

Panos K. Chrysanthis
Dept. of Computer Science
University of Pittsburgh
panos@cs.pitt.edu

***Abstract*—The COVID-19 pandemic poses new challenges in providing *safe* pedestrian navigation information that helps to reduce the risk of severe illness due to the highly contagious nature of the virus. In this paper, we present an innovative system, dubbed *HealthDist*, which utilizes the context (e.g., weather conditions), location (e.g., crowded areas) and user's preferences to support safe mobility. It consists of four modules that allow efficient contact tracing, social distancing, and isolation. *HealthDist*'s modular design reduces the time and resources needed to provide accurate localization for measuring density in common spaces and measuring potential infection exposure, and recommend outdoor and indoor paths satisfying the user's preferences. *HealthDist*'s initial deployment within a university campus demonstrated its capability to provide real time navigation information that reduces the COVID-19 exposure risk while at the same time satisfying the constraints defined by the user.**

***Index Terms*—indoor, outdoor, navigation, path recommendation, graph processing, disability, congestion forecasting**

I. Introduction

On December 31, 2019 a severe pneumonia of relatively unknown cause was reported from Wuhan, China to the World Health Organization (WHO) [1]. COVID-19 is an airborne disease, which is highly contagious with the larger percentage of infected people not exhibiting symptoms. This lead the governments around the world to focus on reducing the spread by using Contact Tracing (CT) and advocating isolation. A number of different CT techniques have been proposed that can automate the contact tracing process and address the challenges of slow tracing procedure requiring a massive human effort. Particularly, the information collected from CT applications is used to compute the risk of the COVID-19 exposure for the contacts based on the context, duration and proximity [2]. Unfortunately, CT systems are not producing the expected results because of low participation rates and a lack of user trust due to privacy concerns [3], [4].

Besides contact tracing, face covering and social distancing can significantly reduce the spread of the COVID-19 disease. Specifically, the wide availability of smartphones, wearable and IoT devices allowed novel social distancing applications

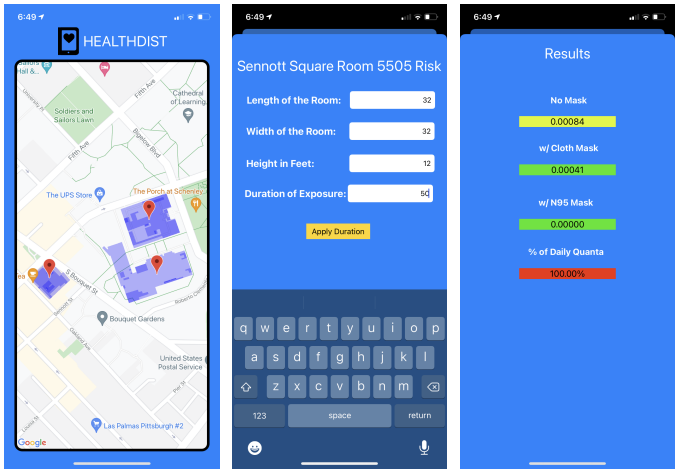

Fig. 1. (left) HealthDist Smart Client application for several University of Pittsburgh campus buildings; (center) user interface for entering information for Sennott Square room 5505; (right) calculated risk of exposure using the parameters provided in the screen.

to warn people to observe social distancing by measuring the distance through inexpensive sensors [5], [6]. Furthermore, new machine learning techniques have been developed to predict the infection risk score and analyze the information for contact tracing [7], [8]. While CT systems share similarity to our own *HealthDist* system with regard to contact tracing, they do not emphasize *proactive* measures such as contact avoidance which are emphasized in *HealthDist*.

In this paper, we present our innovative system, dubbed *HealthDist*, designed as part of the *CovidReduce* project (CovidReduce.org) to implement a holistic approach of *proactive* (contact avoidance) and *retroactive* (contact tracing) functionalities without violating privacy. Beyond COVID-19, *HealthDist* can be used in combating any crowd diseases, such as influenza, which are most commonly spread from an infected person to others through the air by coughs, sneezes, and close personal contact, such as touching or shaking hands.

It utilizes the context (e.g., weather conditions), location (e.g., crowded areas) and user's preferences to recommend *safe* pedestrian paths that go through less congested hallways and corridors, hence decreasing the exposure to the virus causing COVID-19 and reducing the risk of severe illness.

Furthermore, *HealthDist* is designed to serve as a highly accurate *proximity detector* to provide users with information about the distance of nearby individuals, as well as a radar detector to provide information about potential encounters with moving, even out of view, individuals in their vicinity (path/trajectory prediction), and operate as an *infection exposure meter*, counting possible infection Quanta (i.e., infection dose). As a *proximity detector* and *radar detector* the system is able to measure density in a common space area and coordinate access to the common space area based on safety parameters. In all cases, the information to individuals/users is provided in the form of push notifications and public displays.

HealthDist consists of three components, namely *HealthDist Back-end*, *HealthDist Smart Clients*, and *HealthDist BLE Infrastructure* that are implemented by four modules that perform contact avoidance and contact tracing. Its modular design reduces the time and resources needed to provide accurate localization and recommend outdoor and indoor paths satisfying the user's preferences or situation (e.g., got vaccinated or not).

HealthDist's initial deployment within a university campus demonstrated its capability to provide real time navigation assistance with reduced risk of the COVID-19 exposure while at the same time satisfying the constraints defined by the user in terms of accessibility requirements, congestion tolerance, arrival time, and outdoor exposure.

The contributions of this work are summarized as follows:

- We identify the requirements and services of an effective solution in the fight against COVID-19 pandemic that supports both contact avoidance and contact tracing.
- We propose a novel architecture along with the underlying infrastructure that supports safe mobility based on the context, location and user's preferences that reduces personal infection risk from airborne viruses.
- We present a prototype *HealthDist* system and describe two aspects of its experimental evaluation, namely localization accuracy and avoiding congested spaces, when it was deployed in the University of Pittsburgh campus.
- We show the effectiveness of *HealthDist* path recommendation to reduce the risk of COVID-19 exposure by satisfying the user's predefined maximum congestion tolerance and outdoor exposure limit, and identify that the WkNN algorithm offers the best accuracy in a setting with simple, unoptimized BLE deployment.

Section II describes the requirements that a context, location and preference-aware service should provide to reduce the spread of a pandemic like COVID-19. Section III gives an overview of our novel architecture along with the basic infrastructure and smart clients. Our prototype is described in Section IV and the experimental evaluation in Section V. Our concluding remarks are in Section VI.

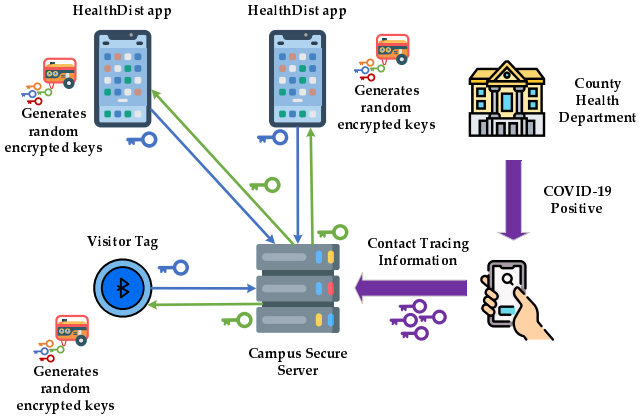

Fig. 2. The HealthDist app has encrypted information to benefit the user that is not transmitted to the university including user contract information, campus route planning, and health IRUs contact tracing history (4 weeks).

II. HealthDist Services and Requirements

The *HealthDist* system consists of three distinct components: (i) *HealthDist Back-end*; (ii) *HealthDist Smart Clients*; and (iii) *HealthDist BLE Infrastructure*. The *HealthDist* Back-end is composed of trusted servers that facilitate the stateless interactions among the smartphones for contact tracing and the stateless computations for contact avoidance.

The *HealthDist Smart Clients* shown in Figure 1 is the collection of the individual smartphones running the *HealthDist* system and the *HealthDist* tag clients supporting their participatory service executed on the trusted server. The *HealthDist* tag clients are necessary for people who want to participate but cannot run the *HealthDist* system on their phones.

The *HealthDist BLE infrastructure* consists of a network of BLE devices (BLE beacons) and other stationary Bluetooth enabled devices (e.g., printers, monitors, desktop servers). A component of the infrastructure layer is the *HealthDist* Logger, which is used to collect fingerprints from the BLE supported buildings that are needed by the BLE & Building Management Server to produce radio maps for localization and navigation.

HealthDist preserves individuals' privacy by recording, storing, and carrying all the necessary computations using the encrypted data on their smartphones. The central server is assumed to be trusted and honest, that is, the server will function as expected and will not disclose any information such as IDs reported when an individual is exposed. The decentralized *HealthDist* design allows individuals to participate in the collective stateless services without revealing their real identity and being tracked by the system. Specifically, the *HealthDist* system generates unique IDs that are updated in a pre-specified random time interval (e.g., 15 minutes) that allows individuals to be anonymous at all times, even while moving through different spaces. Additionally, all the communication and data manipulation use state-of-the-art obfuscation and encryption (e.g., AES-256) to ensure secure data protection and provide strong privacy guarantees in the system illustrated in Figure 2.

A. Online/Proactive services

In order to support visitors and persons who do not have *HealthDist* installed or an available smartphone, and who wish to participate in HealthDist, they can use the trusted server, which acts as a virtual smartphone. The trusted server includes these individuals in the *HealthDist* computation rounds temporarily (as long as the individual wishes) and preserves their privacy as for all other *HealthDist* system users. However, the trusted server is required to be able to directly communicate with these users via email or SMS on their regular phones.

1) Localization Service: Indoor localization of high accuracy is achieved by combining fingerprinting, and Bluetooth Low Energy (BLE) beacon signals.

The fingerprinting is an important part of the localization procedure and requires: (i) BLE placement in the buildings; and (ii) Continuous recording of the BLE beacon devices identification and RSSI (Received Signal Strength Indicator). The BLE placement requires an accurate BLE management subsystem that manages the location, the time of the placement, and the approximated battery level of the BLE devices. When the BLE devices are broadcasting data packets at regular intervals of time, the MAC address and the RSSI can be recorded as a tuple (e.g., <64:e2:50:b4:b3:0f, -78>) and the smartphones can receive the packets in close proximity. All iPhones newer than the iPhone 4 (∼98% market share) and all Android phones with Android 4.3 (∼97.6% market share) and up support BLE localization service. For both types of smartphones, the service requires to access the BLE list and extracts this information to get the predicted location based on the prerecorded radio map. The BLE Management module is responsible for storing the data from the fingerprinting logging procedure in a remote NoSQL database (e.g., MongoDB) and a distributed filesystem (e.g., GlusterFS).

2) Proximity Service: The proximity service is designed in a participatory fashion where all the users are anonymously reporting periodically their location calculated by the indoor localization service on their smartphones. As mentioned above, this is achieved by generating unique, anonymized, and encrypted IDs randomly and periodically by each smartphone to support privacy-preserving trajectory and encounter prediction at a trusted server. The trusted server collects and carries out the proximity stateless detection. That is, the trusted server never stores any localization information which is discarded once each user is notified about the nearest neighbors, highlighting the risk of contact in 3-level distances (safe/green, warning/yellow, dangerous/red) defined by the COVID-19 safety rules [9].

3) Density Service: Reporting density to an individual device or to a bulletin board works in similar fashion as in Proximity Detection with the addition of providing the id or the bounding box of the shared space area. In the same participatory fashion, the system can identify the congested common spaces based on the provided guidelines for social distancing from the CDC [9]. To avoid exposing sensitive room information the floor plans of a building highlight only the common congested areas as shown in Figure 3. The building information is stored using the GeoJSON format that allows us to easily extract the details about the building's floors. In the case of shared space areas (e.g., classrooms and conference rooms) with entrance and exit doors, pairs of safety sensors beam switches configured to detect direction of movement are used to count individuals entering and leaving the common area, providing additional anonymized density information.

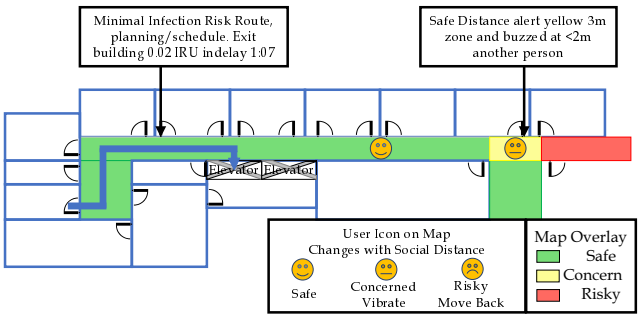

Fig. 3. Infection Risk Routing Scheduling allows the smartphone user to identify the risk for indoor segments.

4) Reservation Service: This is responsible for requesting access to shared spaces and reserving a time slot. A user submits a request to the secure server by providing the id or the bounding box of the shared space area. A request is granted if density requirements are met.

5) Contact Avoidance Service: In this participatory service, individuals may choose to upload their positions and destinations and request path recommendation that avoids congested areas, hence reducing the risk of COVID-19 exposure. In finding an indoor path to recommend, the trusted server utilizes the Density Service as well as a congestion forecasting module. We use machine learning, in particular the long short-term memory (LSTM) models [10], to build a congestion model for each indoor segment (i.e., corridor or hallway) based on historical data and event calendars. Again, individuals interact with the trusted server anonymously, encrypting their sensitive data using the AES keys generated by the smartphones during the initial communication with the central server.

6) Viral Risk Assessment Service: It uses collected data of time, spaces, and activities during the day to calculate the *potential viral load* (PVL), which is the integral of distance to others, volume of the locations, duration of the encounters, activities (i.e., breathing rate), and state of *Personal Protective Equipment* (PPE) in predicted viral exposure of all others that were infected. All calculations are based only for the airborne transmission of the virus and on estimation of a relative metric, called *quanta*, which can be translated to the probability estimates of viral infection. Viral risk assessment calculations take place on users' smartphones and calculated quanta are maintained encrypted on the smartphones as the other health application's data. All the used data calculating quanta are purged daily. Details about these parameters can be found in the web-based Quanta calculator available at CovidReduce.org.

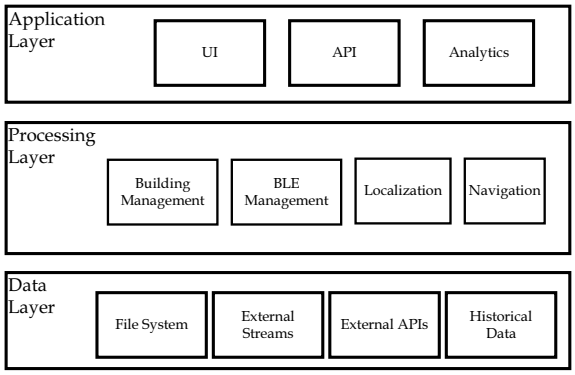

Fig. 4. The HealthDist system builds on top of three conceptual layers from CAPRIO: i) Application Layer; ii) Processing Layer; and iii) Data Layer

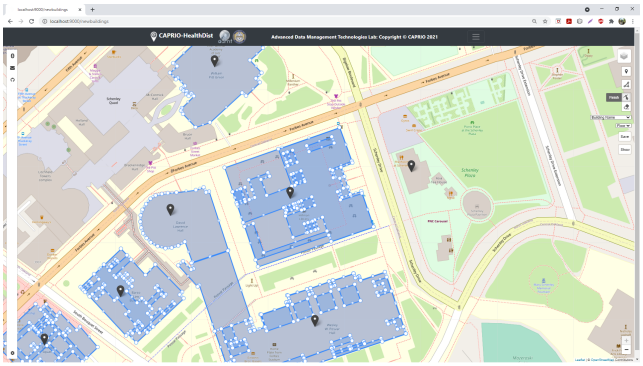

Fig. 5. The Building Management (BM) allows the easy and simple management of buildings

B. *Offline/Retroactive Service*

Contact Tracing Service: The contact tracing service has two detection modes: (i) direct detection; and (ii) indirect detection. In the direct detection, a smartphone is recording the encrypted unique IDs of the users that were in at least 6 feet proximity, per CDC guidelines [9], along with a timestamp. In the indirect mode, the smartphones are recording the trajectories of the users daily and can support similar functionality as in the direct mode. An encrypted list of IDs and the trajectories are stored on the user's smartphone and the data are never uploaded or exported to third-party agencies. Therefore, the user's privacy is preserved. The data can be decayed eventually after the 2-3 weeks considering the recommended quarantine duration. If an individual is found to be a positive COVID-19 case, this individual is expected to notify the *HealthDist* trusted server/agent by uploading the set of their encrypted ID and trajectories. The trusted server will disseminate the list of IDs and trajectories to the *HealthDist* users in both push (broadcasting) and pull (on-demand daily) modes.

The encrypted IDs of the infected individual are used for direct contact tracing by comparing them with the IDs of people who were recorded within the proximity of a user's smartphone. The encrypted trajectories of the infected individual are used for indirect contact tracing by matching them with those recorded on a user's smartphone (utilizing the ideas from the SmartTrace framework [11]). Trajectory matching recognizes cases of individuals who were in the same shared space in a building during a time interval without direct contact. In both direct and indirect contact tracing cases, non-infected individuals do not disclose their own trajectories by carrying out the detection on their own devices. In all cases the identities of infected and non-infected individuals are never disclosed. People without smartphones as well as visitors can use BLE tags, but their trajectories need to be stored at the secure server accessible only by trusted personnel.

III. HealthDist Back-end Architecture

HealthDist builds upon our CAPRIO architecture (Figure 4) and consists of three conceptual layers, the *Data Layer*, *Processing Layer*, and *Application Layer*.

The Data Layer is responsible for managing input data from various data sources such as local or distributed files, data streams, or other external APIs. The Data Layer ensures that any incoming data is formatted properly for the Processing Layer before sending the data to be processed.

The Processing Layer is responsible for the core functionalities and services of the *HealthDist* system such as processing data for path recommendations or performing localization to provide congestion data. This layer is comprised of four modules that support contact tracing, social distancing, and isolation: (i) the *Building Management*; (ii) the *BLE Management*; (iii) *Localization/Radio map generation*; and (iv) the *Navigation Assistant* component. The localization module interacts with the Building Management to retrieve the geometry and the indoor path of a building and with the BLE Management to retrieve the BLE fingerprints and generate a radio map. The Navigation Assistant module utilizes the radio map and the algorithms provided by our *LocationAccuracy* library in combination with the recommended path by our CAPRIO service to provide safe indoor and outdoor navigation. *HealthDist* preserves privacy by using unique, anonymized, and encrypted IDs (based on the Apple/Android SDK) as shown in Figure 2 and discussed in Section II.

The Application Layer acts as the user interface for *HealthDist* and allows for easy development with an open API. This layer utilizes a Leaflet JS map to abstract the complexities of the system from the user. The Application Layer contains two additional components which are integral to the Processing Layer and indoor-path recommendations. The components *Building Management* and *BLE Management* allow the insertion, update, and displaying of BLE devices. The BLE system places BLE devices into a specific building, floor, and corridor that are provided and managed by the Building Management module. These BLE devices are then used as part of a fingerprint in the indoor localization component of the Processing layer which provides input data to the congestion forecasting unit.

A. *Building Management*

The Building Management (BM) is a vital part of the *HealthDist* system and is responsible for storing the geometry

of each floor of all buildings in a way that can provide the information efficiently to build more accurate models [12]. The COVID-19 indoor exposure is related with the indoor congestion and directionality [13]. The BM is designed and implemented over a NoSQL architecture in order to support all the indoor elements (e.g., corridors, rooms, shelves) that affect the propagation of the COVID-19 virus in indoor places.

Figure 5 shows the intuitive user interface of the BM. BM allows the user to depict buildings and their respective corridors, entrances and exits as geometry layers on a map. These geometry layers can then be stored, edited and deleted as required. The purpose of this system is to avoid any discrepancies that exist in building outlines in map tiles and provide a higher level of accuracy during outdoor and indoor path recommendation. The buildings and their respective features such as corridors, exits, etc. are created as GeoJSON objects using Leaflet JS. The GeoJSON format not only supports all the geographic type (e.g., Point, Multipoint, Line, Polyline, MultiPolyline, Polygon and MultiPolygon) in the most efficient way, but is also a user-friendly and lightweight format based on JSON. Leaflet JS is an ideal Javascript library for handling interactive maps of the system due to its simplicity, usability and extensibility through plugins.

The objects created at the user interface are stored in MongoDB documents using the Play framework and Scala programming language. MongoDB can work with semi-structured data, which is greatly beneficial when working with GeoJSON data. It also makes it easier to update data as it grants access to individual fields. This feature grants more flexibility when manipulating pre-existing documents and was helpful when documents grew in complexity with the addition of coordinates. An additional benefit to using MongoDB is GeoJSON specific indexing such as 2dsphere, which is very useful for localization. Play framework provides great NoSQL support such as the Mongo module which is essential for the system. Furthermore, it grants the use of both Scala and Java programming. Scala, on the other hand, provides the combined experience of object-oriented and functional programming, as well as high scalability. It also allows the execution of Java code.

B. BLE Management

The Bluetooth Low Energy (BLE) Management (BLEM) is responsible for storing information about the location, identification and hardware characteristics of each BLE in all of the buildings for maintenance, management, and localization purposes.

Figure 6 shows the intuitive user interface of the BLEM. BLEM handles the creation (placement), editing and retrieval of the virtual forms of the BLE beacon devices. Once placed in designated positions, the data collected from these devices helps in user indoor localization in the Corridor(s) of Building(s). The BLE model shares two attributes with the building and corridor models from the BMS (i.e., building code and floor attributes). These attributes help in assigning the placement of BLE devices to a particular floor of a specific

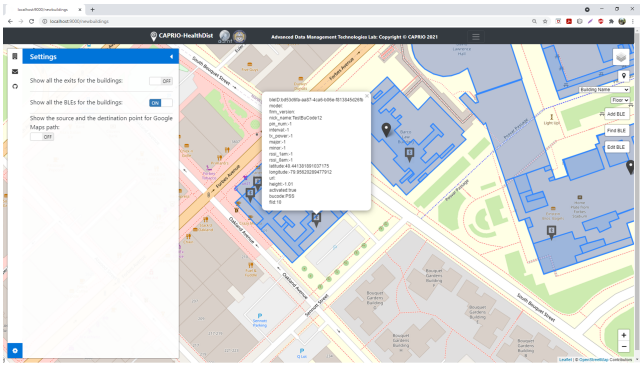

Fig. 6. The BLE Management (BLEM) supports efficient management of the BLE devices using an intuitive map-based interface

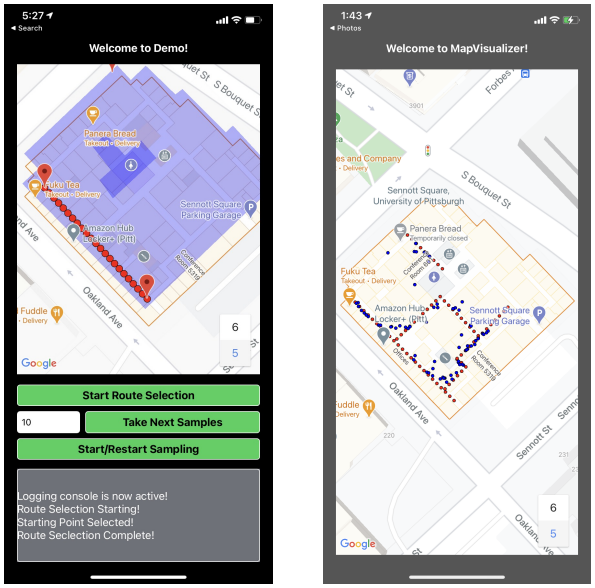

Fig. 7. (left) The BLE logger application, (right) Location Visualization application for the localization

building. The BLE model has 16 other attributes (e.g., BLE ID, Model, Firm Version, Nickname, Pin Number, Interval, Tx Power, Major, Minor, RSSI 1am, RSSI 0am, URL, Latitude, Longitude, Height and Activated). The BLE ID is a unique identifier and can be used for retrieving a particular BLE.

C. Localization & Radio Map generation Module

Localization & Radio Map generation Module interacts with the BLEM to quickly retrieve BLE measurements collected by the BLE logger shown in Figure 7 (left) through BLE fingerprinting. The BLE fingerprints are used to generate radio maps that are stored in MongoDB and can be quickly retrieved using the spatial indexes. The BLE measurements are important for the generation of the radio map of a specific building and floor described in the Section III-B. The functionalities of this module are expressed as a localization library that can be used by any application for localization.

In order to streamline the process of creating a visual representation, analysis of radio maps, and the localization

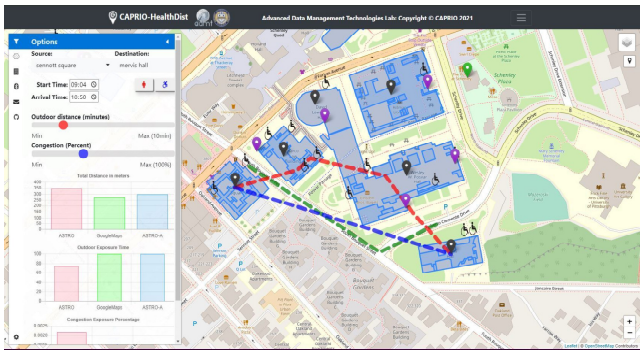

Fig. 8. (top) The HealthDist data exploration user interface was developed on top of our CAPRIO system, which enables the direct comparison between our *CAPRIO* recommended path (blue line), the path from traditional navigation systems, like Google Maps (green line) and the HealthDist-*CAPRIO* v2.0 path (red line), which takes into consideration the accessibility and the congestion. (Left) The HealthDist-*CAPRIO* V2.0 provides the comparison between the three path in terms of distance, outdoor exposure and congestion in bar plot form along with the respective path on the map.

applications, we developed a Swift library that could be simply imported into any iOS application shown in Figure 7 (right) and a Java library based on the Anyplace's [14] open-source code to be used by any Android application. This library includes many structures that are helpful when dealing with BLE radio maps, latitude and longitude coordinates, and the algorithms themselves. Our localization library, named *LocationAccuracy*, supports the use of these structures and algorithms that would normally have to be included in the code of the application. The library acts as the backbone of predicting positions based on given BLE fingerprints to support the navigation of a recommended path.

The *LocationAccuracy* allows the utilization of relatable radio maps for localization, keeps track of geometric coordinates and uses different metric distances based on the application requirements. Additionally, *LocationAccuracy* supports four localization algorithms developed by Anyplace and described in Section V that will produce predictions when passed the required inputs. The initial version of *LocationAccuracy* library can be updated and recompiled into importable archive files.

D. Navigation Assistant

The Navigation Assistant module provides efficient routing using the recommended path from the CAPRIO system and it is constantly updating the location using the *LocationAccuracy* library by requesting real-time information to support safe navigation. Particularly, it interacts with all the modules to retrieve additional information about routing scheduling and avoid high infection risk areas.

IV. Prototype

As mentioned earlier, *HealthDist* builds upon our *CAPRIO* architecture and the current *HealthDist* prototype is an extension of the *CAPRIO* prototype that integrates the four modules of the Processing Layer described above. The prototype is deployed, i.e., supports partial navigation involving nine buildings of the University of Pittsburgh main campus.

The prototype of *CAPRIO* incorporates an interactive map and integrates several graph techniques in the back-end, which was developed using Play Framework 2.7[1] and MongoDB. The *CAPRIO v2.0* web interface is implemented in HTML5/CSS3 along with extensive usage of Leaflet[2] and Cytoscape.js[3].

An illustrative path exploration interface is shown in Figure 8. We have implemented a query sidebar that allows the user to execute a variety of template queries. The query sidebar has three main tabs: (i) the options tab that enables the user to choose the source, the destination and the accessibility of the recommended path along with its outdoor exposure/distance and the congestion preference, shown in Figure 8 (sidebar); (ii) the graph tab that animates the path using a graph visualization to provide visually the algorithms and techniques behind the paths; and (iii) the settings tab that activates/deactivates elements on the main user interface.

V. Experiments

This section presents an experimental evaluation of our localization and route recommendation methods. We start out with the experimental methodology and setup, followed by two experiments. This section also provides details regarding the algorithms, metrics, and datasets used for evaluating the performance of the proposed approach.

1) Testbed: Our evaluation is carried out on a dedicated Windows 10 server. The server is featuring 12GB of RAM with 4 Cores (@ 2.90GHz), a 500 GB SSD and a 750 GB HDD.

Algorithms: We are comparing 4 localization algorithms. In our experimental evaluation we are using k = 4:

(i) **UkNN** (Unweighted k-Nearest Neighbors) considers only the k points with the shortest signal distance and the average of the coordinates of k points can be used as the estimate of the user's location.

(ii) **WkNN** Weighted k-Nearest Neighbors is very similar to the unweighted version of k-Nearest neighbors. The user's location is computed based on the weighted average rather than the average.

(iii) **PMMSE** (Probabilistic Minimum Mean Square Error) uses probabilistic models to provide a natural way of handling uncertainty and errors in signal power measurements and determines the user's location similarly with Probabilistic Maximum A Posteriori (MAP) Algorithm.

(iv) **WPMME** (Weighted Probabilistic Minimum Mean Square Error) expands upon PMMSE with the difference that the normalized probability affects the estimation of the user's location.

We are also comparing two path recommendation algorithms that are used in our *HealthDist* system:

(i) **Dijkstra**, which is a modified and optimized version of Dijkstra using a priority queue and early termination with additional constraints;

[1]Play Framework: https://www.playframework.com/
[2]Leaflet: https://leafletjs.com/
[3]Cytoscape.js: http://js.cytoscape.org/

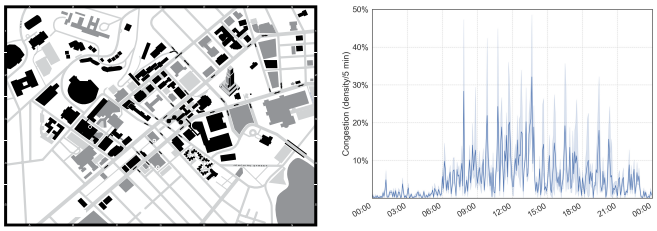

Fig. 9. Dataset for the buildings in University of Pittsburgh campus and its average congestion for one day.

(ii) **CBFS (Closest Building First Search)**, which is an algorithm where the closest building is always selected first.

Datasets: We are using a realistic dataset of the University of Pittsburgh campus, coined **PITT**. It consists of 9 buildings with each building having 2 to 6 doors (3 on average) and up to 582 corridor cells (126 on average). The average door-to-door corridor length is 69 meters.

We used camera analysis [15] on a 2-hour session and extrapolated the congestion data using the University of Pittsburgh Fall 2019 schedule. Then, we generate congestion data based on the schedule and a walking speed of 1.4 m/s [16], [17] for a period of 6 months. Figure 9 shows the map of the Pitt campus and the average generated congestion density (number of people over corridor capacity) of all buildings on July 19, 2019.

Metrics: We evaluate the performance of Dijkstra and CBFS using the metrics: (i) Average error in meters (m), (ii) Maximum congestion, which is the maximum density observed across all indoor paths in an area of 3 square meters; (iii) Average congestion, which is the average density observed across all indoor paths; and (iv) Response time, which measures the execution time using the average of 100 consecutive runs in milliseconds.

2) Experimental Results: Due to space limitations, we report the results of only two experiments, the first experiment we carried out to select the localization algorithm to be used in *HealthDist* and the second experiment to assert the effectiveness of the current path finding algorithm that considers congestion.

Comparison of Localization algorithms: Figure 10 (top) shows that WkNN and UkNN have similar performance with under six meters error and WkNN slightly better. WkNN outperforms both PMMSE and WPMMSE that report seven meters error. Based on these results, we have adopted WkNN and we are currently trying to optimize it. Figure 10 (bottom) shows the error for every single point in the collected dataset verifying our initial hypothesis that in some isolated predictions there is a recording error.

Path Effectiveness: In the absence of any user's preference, such as outdoor exposure time limit and congestion tolerance (*NO LIMITS*), *Dijkstra* incurs zero maximum and average congestion for PITT dataset with the path being totally outdoors

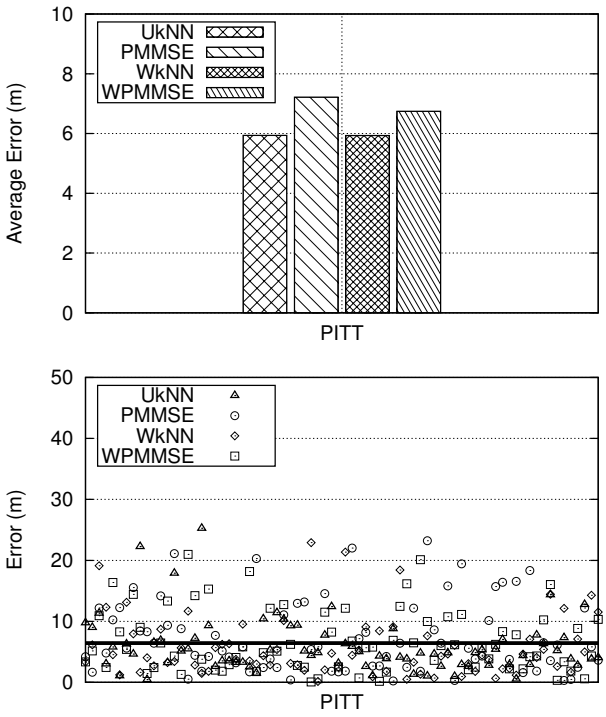

Fig. 10. *Examining the accuracy of the localization algorithms for one building in University of Pittsburgh campus with 13 BLE devices placed 20 feet apart.*

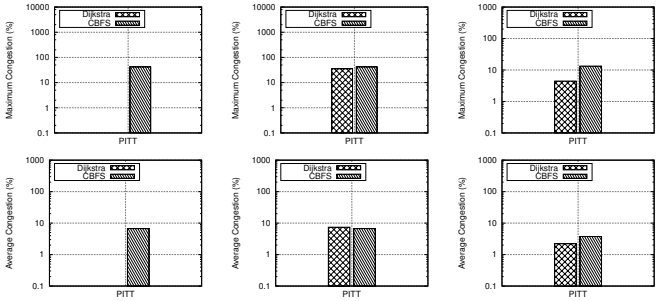

Fig. 11. *Examining the maximum congestion (top), and the average congestion (bottom) in percentage for finding the longest path in the PITT dataset without any preferences defined (left), with outdoor exposure time limit of 300 seconds (center), with outdoor exposure time limit of 300 seconds and 15% congestion tolerance limit (right).*

(shown in Figure 11 top and bottom left). In contrast, *CBFS* spends more than 42% maximum and 6% average congestion. This happens because its strategy is to visit the closest building first irrespective of the presence or absence of constraints, therefore increasing the maximum and average congestion.

With an outdoor exposure time limit of 300 seconds (*O LIMIT*) as shown in Figure 11 top and bottom center, *Dijkstra* incurs 35% maximum and 7% average congestion for PITT dataset due to the outdoor exposure limit preference. *CBFS* spends more than 42% maximum and 6% average congestion.

With an outdoor exposure time limit of 300 seconds and 15% congestion tolerance limit (*O/C LIMIT*) as shown in

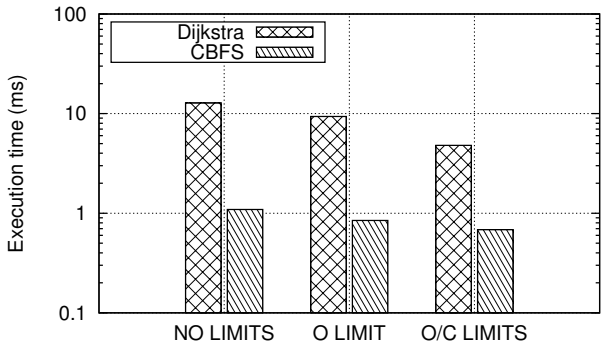

Fig. 12. *Examining the execution time for finding the longest path in the PITT dataset without any preferences defined (left), with outdoor exposure time limit of 300 seconds (center), with outdoor exposure time limit of 300 seconds and 15% congestion tolerance limit (right).*

Figure 11 top and bottom right, *Dijkstra* incurs 4% maximum and 2% average congestion for PITT dataset due to the new limit preferences. *CBFS* spends more than 13% maximum and 4% average congestion.

Figure 12 shows that for all three cases (i.e., no limits, with outdoor exposure limit, and with outdoor exposure and congestion limit) the *CBFS* is faster by one order of magnitude. Based on these results, we are currently adopting *CBFS* in our proposed system and we are currently exploring an A* variation that optimizes the response time of Dijkstra.

VI. Conclusions and Future Work

In this paper we present *HealthDist*, a novel system that supports safe navigation based on the context, location and user's preferences. *HealthDist* combines state-of-the-art systems, software programs, algorithms and easy to set-up infrastructure to provide an alternative and complementary solution to the contact-tracing only systems. Our experimental results show that *HealthDist* can reduce the risk of COVID-19 exposure by satisfying the user's predefined maximum congestion tolerance and outdoor exposure limit. We also found that the WkNN algorithm offers the best accuracy in a quick and simple (unoptimized) BLE deployment, which is often the case during field/pilot deployment and no retrofitting.

In the future, we aim to enhance the congestion modeling process to further improve the forecasting accuracy and efficiency of the models by exploring new Machine Learning models. We also plan to enhance our system with real-time information using crowdsourcing and sensing devices to achieve higher accuracy. Currently the fingerprint mapping requires a human to manually make any environmental changes, we plan to enhance our system by making these updates a continuous, online process. We aim to enhance the experiments by conducting a real deployment of *HealthDist* and examining potential impact on users' behavior. Lastly, we plan to enhance the privacy of our system by considering a stronger threat model that does not assume a trusted central server.

Acknowledgment

We thank Sudhir Pathak and David Busch for the useful discussion during the brainstorming and planning phase of *HealthDist*. Part of BLEM was developed by Robbie Kline as his BS capstone project. Daniel Cohen worked on the CAPRIO prototype while being a research intern during his Senior year.

References

[1] D. Zeinalipour-Yazti and C. Claramunt, "Covid-19 mobile contact tracing apps (mcta): A digital vaccine or a privacy demolition?" in *2020 21st IEEE International Conference on Mobile Data Management (MDM)*, 2020, pp. 1–4.

[2] N. Ahmed, R. A. Michelin, W. Xue, S. Ruj, R. Malaney, S. S. Kanhere, A. Seneviratne, W. Hu, H. Janicke, and S. K. Jha, "A survey of covid-19 contact tracing apps," *IEEE Access*, vol. 8, pp. 134 577–134 601, 2020.

[3] M. F. Mokbel, S. Abbar, and R. Stanojevic, "Contact tracing: Beyond the apps," 2020.

[4] L. Xiong, C. Shahabi, Y. Da, R. Ahuja, V. Hertzberg, L. Waller, X. Jiang, and A. Franklin, "React: Real-time contact tracing and risk monitoring using privacy-enhanced mobile tracking," *SIGSPATIAL Special*, vol. 12, no. 2, p. 3–14, Oct. 2020.

[5] Y. Tachibana and N. Segawa, *Physical Distance Monitoring System for COVID-19 Using Raspberry Pi and a Monocular Camera: Poster Abstract*, 2020, p. 772–773.

[6] S. Bian, B. Zhou, H. Bello, and P. Lukowicz, "A wearable magnetic field based proximity sensing system for monitoring covid-19 social distancing," in *Proceedings of the 2020 International Symposium on Wearable Computers*, ser. ISWC '20, 2020, p. 22–26.

[7] M. Ekpenyong, I. Udo, F.-M. Uzoka, and K. Attai, "A spatio-graphnet model for real-time contact tracing of covid-19 infection in resource limited settings," in *Proceedings of the 4th International Conference on Medical and Health Informatics*, ser. ICMHI 2020, 2020, p. 208–217.

[8] R. Agarwal and A. Banerjee, *Infection Risk Score: Identifying the Risk of Infection Propagation Based on Human Contact*, 2020, p. 1–10.

[9] Centers for Disease Control and Prevention (CDC), "Social distancing guidelines," 2021. [Online]. Available: https://www.cdc.gov/coronavirus/2019-ncov/prevent-getting-sick/social-distancing.html

[10] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Comput.*, vol. 9, no. 8, p. 1735–1780, Nov. 1997.

[11] D. Zeinalipour-Yazti, C. Laoudias, C. Costa, M. Vlachos, M. I. Andreou, and D. Gunopulos, "Crowdsourced trace similarity with smartphones," *IEEE Transactions on Knowledge and Data Engineering*, vol. 25, no. 6, pp. 1240–1253, 2013.

[12] V. Vuorinen, M. Aarnio, M. Alava, V. Alopaeus, N. Atanasova, M. Auvinen, N. Balasubramanian, H. Bordbar, P. Erästö, R. Grande, N. Hayward, A. Hellsten, S. Hostikka, J. Hokkanen, O. Kaario, A. Karvinen, I. Kivistö, M. Korhonen, R. Kosonen, J. Kuusela, S. Lestinen, E. Laurila, H. J. Nieminen, P. Peltonen, J. Pokki, A. Puisto, P. Råback, H. Salmenjoki, T. Sironen, and M. Österberg, "Modelling aerosol transport and virus exposure with numerical simulations in relation to sars-cov-2 transmission by inhalation indoors," *Safety Science*, vol. 130, p. 104866, 2020.

[13] V. Romero, W. D. Stone, and J. D. Ford, "Covid-19 indoor exposure levels: An analysis of foot traffic scenarios within an academic building," *Transportation Research Interdisciplinary Perspectives*, vol. 7, p. 100185, 2020.

[14] D. Zeinalipour-Yazti, C. Laoudias, K. Georgiou, and G. Chatzimilioudis, "Internet-based indoor navigation services," *IEEE Internet Computing*, vol. 21, no. 04, pp. 54–63, jul 2017.

[15] C. Feliciani and K. Nishinari, "Measurement of congestion and intrinsic risk in pedestrian crowds," *Transportation Research Part C: Emerging Technologies*, vol. 91, pp. 124 – 155, 2018.

[16] R. W. Bohannon and A. Williams Andrews, "Normal walking speed: a descriptive meta-analysis," *Physiotherapy*, vol. 97, no. 3, pp. 182 – 189, 2011.

[17] R. L. Knoblauch, M. T. Pietrucha, and M. Nitzburg, "Field studies of pedestrian walking speed and start-up time," *Transportation Research Record*, vol. 1538, no. 1, pp. 27–38, 1996.