Date of publication xxxx 00, 0000, date of current version xxxx 00, 0000.

Digital Object Identifier 10.1109/ACCESS.2017.DOI

Wi-Fi and Bluetooth Contact Tracing Without User Intervention

BROSNAN YUEN[1], YIFENG BIE[1], DUNCAN CAIRNS[1], GEOFFREY HARPER[1], JASON XU[1], CHARLES CHANG[1], XIAODAI DONG[1] (Senior Member, IEEE), and TAO LU[1](Member, IEEE)

[1]Department of Electrical and Computer Engineering, University of Victoria, Victoria, BC, Canada (Emails: {brosnany, xdong, taolu}@uvic.ca)

Corresponding author: Xiaodai Dong (e-mail: xdong@ece.uvic.ca) and Tao Lu (e-mail: taolu@ece.uvic.ca).

This work was supported in part by the Nature Science and Engineering Research Council of Canada (NSERC) Discovery (Grant No. RGPIN-2020-05938), and Threat Reduction Agency (DTRA) Thrust Area 7, Topic G18 (Grant No.GRANT12500317), NSERC Grant 520198, Fortinet Research under Contract 05484 and NVidia under GPU Grant program.

ABSTRACT

Previous contact tracing systems required the users to perform many manual actions, such as installing smartphone applications, joining wireless networks, or carrying custom user devices. This increases the barrier to entry and lowers the user adoption rate. As a result, the contact tracing effectiveness is reduced. Unlike the systems above, we propose a new privacy preserving Wi-Fi and Bluetooth (BLE) contact tracing system that does not require smartphone applications, joining wireless networks, or custom user devices. Our specially built routers seamlessly track smartphones, laptops, smartwatches, BLE headphones, and tablets without any user action, but do not trace user identity. Mapping between devices and users is only carried out for confirmed cases and suspected contacts. Moreover, we can track the absolute positions of user devices within 1.0 m due to using bidirectional long short-term memory neural networks that are trained with data pre-collected by an autonomous robot. This allows public health authorities to track indirect droplet and surface transmissions that other contact tracing systems often overlook.

INDEX TERMS Contact tracing, Received signal strength indicator (RSSI), Round trip time (RTT), Fine time measurement (FTM), Wi-Fi indoor localization, Bluetooth indoor localization

I. INTRODUCTION

When a new outbreak appears with unknown pathogens, vaccines and treatments are not available immediately to reduce the spread of the disease. Therefore, governments and public health agencies use extensive disease testing to identify infected individuals. However, testing the entire population is inefficient because of the limited testing capacity, false negative cases, and the associated costs. Contact tracing has been developed to make efficient use of the limited testing resources, where the closest contacts of the confirmed cases or symptomatic cases are tested and isolated.

A. CONTACT TRACING

Contact tracing is difficult because super-spreaders could infect thousands of people a day [1] and exponentially increase the number of people in the contact tracing list. Traditionally, contact tracing has been done by hand, where the authorities interview each confirmed case to get the contacts and visited places. Afterwards, suspected cases are isolated and tested. Symptomatic cases and high exposure cases get a higher priority in testing. With a high enough contact tracing efficiency, diseases can be locally contained and sometimes be eradicated [2]. However, performing contact tracing manually is very inefficient because the infected people might forget who they met and where they visited. Staff shortages, incorrect training, and slow turnaround times can also cause inefficient contact tracing.

Many countries have moved to automated means of contact tracing [3]–[5] via smartphones, cameras, custom tracking devices, or genome sequencing. Cameras can be used in-conjunction with facial recognition software to track individual people. Researchers collected a database of faces and applied a convolutional neural network (CNN) to classify the presences of the people in the database [6]. They are able to perform contact tracing via a web interface. Instead of only classifying faces, other researchers have used multiple cameras to track movements in real time [7]. Furthermore, they can determine the actual paths of the confirmed cases for contact tracing.

Genome sequencing enables contact tracing without interviewing patients or requiring tracking devices. This particularly useful for incapacitated or unidentified patients.

Jennifer L. Gardy et al. [8] applied hierarchical clustering to sequenced genomes in order to create a genome tree of a tuberculosis outbreak. Moreover, the genome tree perfectly matches the contact traced social network created from patient questionnaires. The main disadvantage of genome sequencing is the genome tree can only be created after the patients are infected.

On the other hand, smartphones are readily available and can be used for tracking the movements of individuals. Thus, many governments, public health agencies, and software companies have implemented smartphone applications for contact tracing. The Singaporean government released one of the first contact tracing applications for COVID-19 [9]. Each smartphone application broadcasts Bluetooth Low Energy (BLE) exposure notification packets containing temporary IDs of the users. Furthermore, each smartphone receives exposure notifications from all other smartphones and checks the received temporary IDs against a database of confirmed cases. If the temporary ID is in the database of confirmed cases, then the application warns the user about an exposure. Similarly, Apple and Google have developed their own contact tracing system using BLE [10], [11], where they built contact tracing functionality into iOS and Android operating systems. This allows them to do contact tracing on a scale of multiple countries, which is far greater than any other research study. On the other hand, researchers have developed DigitalPPE [12], a wearable BLE smartwatch, that tracks social interacts between people. DigitalPPE gives a vibration warning if two people get too close and records the IDs of the smartwatches with the relative distance. T. Shelby et al. [13] performed two BLE contact tracing studies: one study using a smartphone application and the other study using external BLE tags on the user. The custom BLE tags had a higher accuracy compared to the smartphone application because the BLE tags had a higher transmit rate and power. Also, other researchers have relentlessly applied BLE for contact tracing [14]–[21].

Alternatively, T. Yasaka et al. [22] used QR codes for tracking social gatherings between groups of people. The host of the social gathering creates a QR code using the application, and the participants scan the QR code to build a time series graph. When a user indicates a positive test result, all users within 3 traversals of the time series graph are notified. A few more research papers have used the QR code approach [23]–[25]. Moreover, the smartphones' GPS can be used to track users in the outdoor environments [26]. This would provide a higher position accuracy than BLE and QR codes.

Wi-Fi can be a useful tool for localization and contact tracing. A. Trivedi et al. [27] developed a Wi-Fi based contact tracing system without the need to install an application onto the smartphone. They used the access points (APs) of two universities to collect packets from smartphones, where the user's trajectory is built using the closest APs. Furthermore, a graph search algorithm takes the user's trajectory and produces a location and proximity report of the exposed users. Other research groups have used Wi-Fi based smartphone applications [28] to capture beacon frames from nearby APs and upload the data to the cloud. This allows the authorities to track the visited places and the positions of the confirmed cases. Moreover, the lifespan of the disease can be known due to the recorded timestamps of the beacon frames.

B. INDOOR LOCALIZATION

Localization is fundamental to contact tracing, and it has two major categories: outdoor and indoor localization. Out of all the outdoor localization methods, Global Positioning System (GPS) is the most popular and is robust against signal interference and jamming [29]. However, GPS requires direct line-of-sight (LoS) between the satellites and the handset, which is unsuitable for indoor localization.

Indoor localization has drawn more attention in the industry for its wide variety of use cases, such as autonomous indoor vehicles (AIVs) [30], unmanned aerial vehicles (UAVs) [31], home automation, and smart buildings [32]. Radio Frequency (RF) waves penetrate materials like tables and walls, making RF-based indoor localization the most adopted solution. Moreover, RF performs better than other methods [33]. RF-based systems employ mobile phones for capturing wireless parameters such as angle of arrival (AOA), time of arrival (TOA), and received signal strength indication (RSSI). There are two types of RF based localization methods: ranging and trilateration/triangulation, and fingerprinting. The first method requires deploying known anchor nodes with coordinate information and synchronization among nodes, while the second method does not. In this paper, we use the wireless fingerprinting approach where the RF parameters act as fingerprints for positioning. With the help of machine learning, the average localization error of fingerprinting is around 1 m [34], [35].

Our interests lie in estimating the positions of people to determine COVID-19 exposures. As a result, indoor localization is more useful than outdoor localization due to indoor environments having a higher infection rate [36]. Moreover, indoor localization is extremely helpful for tracking COVID-19 outbreaks in complex environments such as supermarkets and airports.

C. FEATURES OF THE PROPOSED CONTACT TRACING SYSTEM

Every contact tracing system has its own unique features and advantages, as shown in Table 1. Camera contact tracing systems [6] do not require any smartphone applications, wireless network connections, and external user devices. Moreover, they have an accuracy of 0.5 m and can track droplet and surface transmissions. However, setting up multiple cameras per building and a video processing system is extremely costly. Similar to the camera contact tracing system, genome sequencing [8] is highly accurate and precise. However, it requires viral samples from each user and processing each sample is expensive.

TABLE 1: Feature Comparison of Indoor Contact Tracing Systems

Contact Tracing System	Tracking Method	Requires Smartphone Applications ?	Requires Wireless Network Connections ?	Requires External User Devices ?	Tracks Droplet and Surface Exposures ?	Accuracy	Cost
N. Nanthini et al. [6]	Camera	No	No	No	Yes	Absolute Position Within 0.5 m	1 Camera per Room
Jennifer L. Gardy et al. [8]	Genome Sequencing	No	No	No	Yes	1% Error Rate	>$400 USD per Sequence
TraceTogether [9]	BLE	Yes	Yes	No	No	Relative Position Within 1.0 m	Free
Apple and Google Exposure Notification [10], [11]	BLE	No, Requires Manual Activation	Yes	No	No	Relative Position Within 1.0 m	Free
DigitalPPE [12]	BLE	No	No	Yes	No	Relative Position Within 1.0 m	1 Wearable per Person
T. Yasaka et al. [22]	QR codes	Yes	Yes	No	No	Requires Users to Scan QR Codes	Free
A. Trivedi et al. [27]	Wi-Fi	No	Yes	No	No	Position Within the Room	>$340 USD per Router
vContact [28]	Wi-Fi	Optional app automates contact tracing*	Optional app automates contact tracing*	No	Yes	Absolute Position Within 2.0 m	Free*
Proposed Contact Tracing System	**Wi-Fi and BLE**	**No**	**No**	**No**	**Yes**	**Absolute Position Within 1.0 m**	**$30 USD per Router**

* Note: If the user does not install the optional smartphone application, then a medical personnel manually performs contact tracing by revisiting every location the user has been to.

BLE contact tracing systems are cheap and easy to set up. However, they usually require the user to manually install smartphone applications [9] or manually activate exposure notifications in the settings [10], [11]. This results in a low participation rate and decreases the accuracy of contact tracing. Furthermore, those systems only record the relative positions of the users, of which are highly ineffective in tracking droplet and surface transmissions. Using custom BLE smartwatches [12] or tags [13] eliminates the need for smartphone applications, but they only record relative positions and have the exact same problems.

A few Wi-Fi contact tracing systems [27] do not require the user to install smartphone applications. Instead, the routers record the positions of the smartphones, whenever the user manually logs into the wireless network. This approach has low position accuracy due to users getting disconnected or logging out. Furthermore, the RSSI ranking system has location ambiguity due to multiple positions having the same RSSI ranking. As a result, they can not determine if a user is within 2.0 m of another user. Moreover, their system is expensive due to them using Cisco and HP/Aruba equipment that cost >$340 USD per router.

Unlike the previous systems in Table 1, we propose a new privacy preserving Wi-Fi and BLE indoor contact tracing system that does not require the users to perform any actions. Specifically, the users do not need to install any smartphone applications. The users do not need to connect to any wireless networks, which improves localization accuracy due to eliminating wireless network disconnects and users logging out. Instead, we use custom designed ESP32C3 routers to capture Wi-Fi and BLE packets emitted from the wireless devices. The overall system contains four modules: an autonomous robot for site survey, a BiLSTM network for trajectory prediction, WiFi routers designed with special features to collect sufficient RF data and pre-process the data, and graph based contact tracing algorithm. In each module, there are innovations in design solution and practical implementation, as detailed in the next sections. This system allows us to track smartphones, smartwatches, tablets, and laptops of users seamlessly in the background. Although user tracking and device tracking are used interchangeably throughout the paper, the system does not obtain user identity for privacy purpose but only trace device WiFi interface MAC addresses. Our contact tracing system also tracks droplet and surface transmissions due to our neural networks providing absolute positions. Subsequently, indirect or delayed infections can be tracked even-though the infected individual has left the area multiple days ago. As for localization accuracy, our contact tracing system has an average error of 1.0 m, which is similar to the other BLE and Wi-Fi contact tracing systems. However, the camera and genome sequencing methods have higher localization accuracy at a cost of much more expen-

sive equipment.

The paper is organized as follows. Section II is a big picture overview of the proposed contact tracing system. The site survey is conducted in Section III, while the data processing is shown in Section IV. The actual contact tracing algorithm is depicted in Section V. Section VI shows the results and discussions of identifying unique mobile devices, localization performance, and contact tracing. A conclusion is presented in Section VII.

II. OVERVIEW OF THE PROPOSED CONTACT TRACING SYSTEM

Fig. 1 shows the overview of the proposed contact tracing system. It consists of four components: 1) An autonomous robot for site survey to generate a location-fingerprint database; 2) A BiLSTM neural network trained by the site survey dataset for user trajectory prediction; 3) WiFi routers for capturing packets without user action in prediction, testing and training stages; 4) Contact tracing algorithm and engine based on the localization data.

We propose to use a bidirectional long short term memory (BiLSTM) neural networks to predict the trajectories of mobile devices. The BiLSTM requires many datasets such as the training dataset, the testing dataset, and the production dataset. The training dataset is used to train the BiLSTM, while the testing dataset measures the accuracy of the BiLSTM. Moreover, the production dataset is the real life dataset that only contains input features without any labels.

Wireless fingerprinting for localization does not need to install known anchor nodes but does need to have a location-fingerprint database of a site. This site survey if done manually is very laborious. The purpose of the Turtlebot3 site survey is to obtain the training dataset and the testing dataset using a robot. The Turtlebot3 executes autonomous site surveys by meticulously visiting all positions on the floor. A smartphone is mounted on the robot, and it broadcasts wireless packets while moving in order to simulate mobile device trajectories. On the other hand, the ESP32C3 routers capture Wi-Fi and BLE packets for the training dataset, the testing dataset, and the production dataset. For the production dataset, the Turtlebot3 is not involved, and the routers directly capture packets from the users without the users needing to perform any action. Afterwards, the packets' transmit power (TX power), received signal strength indication (RSSI), and time of flight (ToF) are used to predict the user trajectories.

Finally, we design a graph based contact tracing algorithm to build a social contact graph. Every user is assigned to a unique node on the graph. For every intersection between the trajectory of a confirmed case and the trajectory of a user, we add an edge that connects the node of the confirmed case to the node of the user. After repeating this process multiple times, a graph of the suspected cases is displayed together with their trajectories.

In the next sections, each component of the system is described in details.

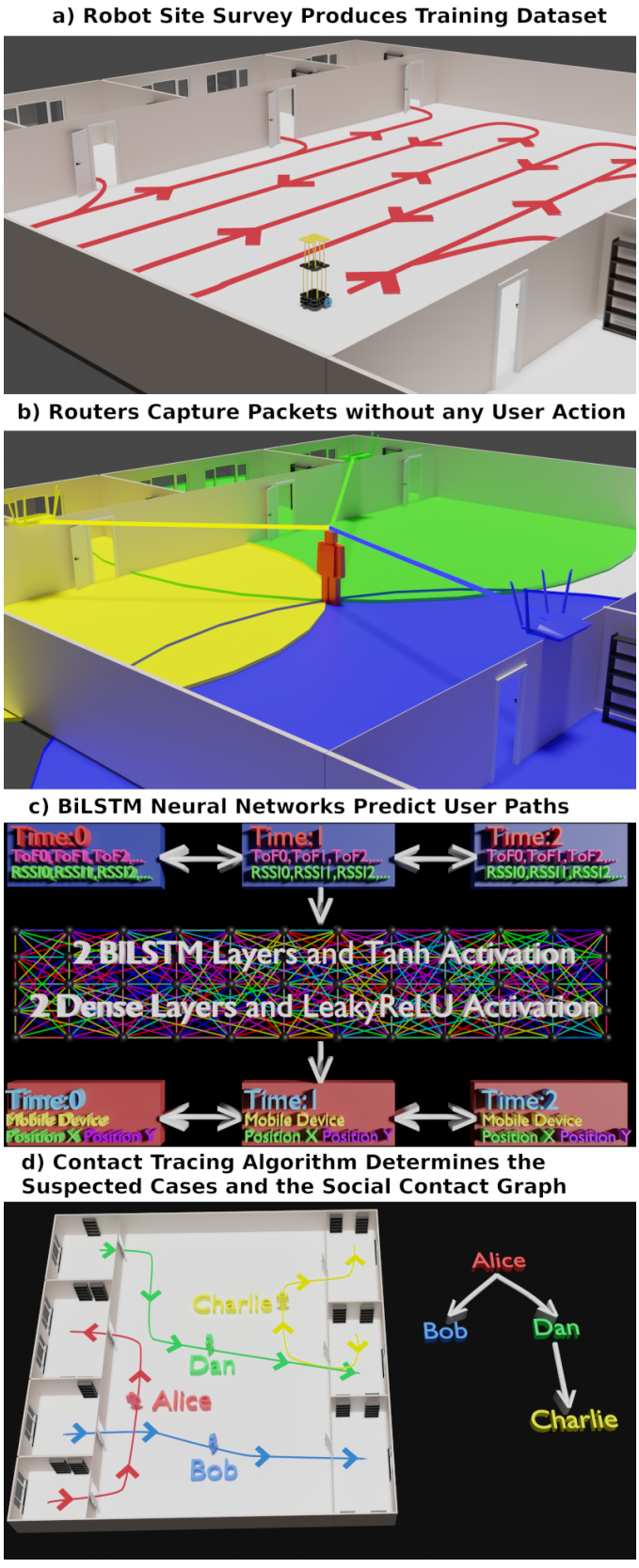

FIGURE 1: Overview of the proposed contact tracing system.

III. TURTLEBOT3 FOR SITE SURVEY

Typically, mobile devices transmit many Wi-Fi and BLE packets as they move around the building. By implementing packet sniffing on the router side, mobile devices can be tracked throughout the day. However, the localization algorithms require large amounts of training and testing data. Measurement of the training data is done in the form of a site survey, where a mobile device transmits packets at every position and the signal information is recorded at the router side.

Collecting data by hand is extremely tedious and introduces position errors. Instead, we built a custom Turtlebot3 [37] for the site survey. The Turtlebot3 continuously transmits packets to the routers, while visiting every position in the building. The original Turtlebot3 has a height of 19 cm, which is too short for the height of a smartphone on a table or in a user's pocket. A platform is added to the custom Turtlebot3 in order to increase the smartphone's height to 75 cm. Moreover, the custom Turtlebot3 is also equipped with RPLIDAR A2, Intel D415 RGBD camera, Nvidia Jetson TX2, and Raspberry Pi 3.

A. ROBOT OPERATING SYSTEM 2

Robot Operating System 2 (ROS2) [38] is an open-source robotics framework that collects sensor information, executes data processing, implements inter-process communications, and allows real-time control. ROS2 has four main concepts: nodes, topics, services, and actions. Nodes are individual processes, of which execute a singular task like collecting sensor data or filtering information. Nodes can commence one way communications with other nodes by publishing messages to topics. All nodes that subscribe to a specific topic receive the same messages. Unlike topics, services are a two-way communications channel. Nodes can send service requests and receive service responses once the specific operation is completed. Actions are an extension of services, where the nodes receive periodic feedback status messages instead of not receiving feedback messages.

The RPLIDAR A2 is a 2D laser ranging device that measures the distances to the nearest opaque objects. It is a 360° LIDAR that completes 1 revolution every 0.1 seconds. The 360° scans are divided into 360 angle intervals. For each angle interval, the RPLIDAR A2 returns a distance value. The laser scans feed into ROS2 SLAM_toolbox, of which it produces a 2D grid map and publishes the transform from map to odometry (odom). It essentially determines the position and orientation of the Turtlebot3. On the other hand, the Intel D415 RGBD is used for obstacle avoidance. The D415 produces a RGBD point cloud at 720p 30 frames per second (FPS). Camera sensors have false positive readings, where the sensor outputs a ghost point in the absence of objects. In order to eliminate the false positives, multiple RGBD point cloud frames are joined together and are uniformly decimated. Afterwards, the point cloud is organized into clusters, where the cluster centres and standard deviations are calculated. If a point is 1 standard deviation away from the cluster centre, then it is removed.

B. DETECTING OBSTACLES IN THE LOCAL MAP

In order for the routers to collect data, the Turtlebot3 needs to visit every position on the floor and stand there for a few minutes. To accomplish that, we create an autonomous navigation algorithm that is able to avoid static objects and moving obstacles in order to operate in various buildings and environments. There are 4 main steps in Algorithm 1 : detecting obstacles in the local map, finding the start positions of the paths, generating the best path using the start positions, and moving along the planned path.

Before planing a path and moving along it, the Turtlebot3 needs to determine the obstacles in the local vicinity. The function constructCostmap retrieves the odometry, 2D laser scans (LaserScans), and 3D point cloud (PointCloud) to construct a 2D occupancy grid (Costmap). Each pixel in the occupancy grid has a status of occupied, free space, or unknown depending on the sensor information. This helps the robot to avoid obstacles, which are labelled as occupied.

Another function, detectObstacles determines if an input position is near any obstacles in the costmap. It checks every occupied position in the costmap to determine if any of them are closer to the input position than the robot radius. If there is an obstacle closer than the robot radius, then the function indicates the presence of an obstacle by returning HasObstacles, otherwise there are no obstacles and the function returns NoObstacles.

C. FINDING THE START POSITIONS OF THE PATHS

Prior to generating new paths, we must first select the start positions of the paths. As shown in the function findStartPos, rays are created starting at the Turtlebot3's current position (Odom) and ending at the occupied positions in the costmap. For each individual ray, it is checked against the costmap for potential obstacles. If the ray has an obstacle in its path, then the ray is discarded. Afterwards, a random position is selected from the ray's line and is appended to the list of start positions (StartPositionList). The list of start positions represent candidate positions for generating the best path.

D. GENERATING THE BEST PATH USING THE START POSITIONS

After finding the start positions, we use the function createPath to generate new paths and score them for finding the best path. One of the inputs to the function is a list of previously visited positions (VisitedPositionList), of which is used to avoid visiting the same locations. Subsequently, the best path (BestPath) and best path score (BestPathScore) are initialized with the worst possible path. A low path score indicates an undesirable path, while a high path score indicates a desirable path. Afterwards, the algorithm generates every possible combination of candidate path starting from the start position list and ending at an obstacle in the costmap. The candidate path is a straight line that has evenly spaced points

Algorithm 1: Path Planning and Autonomous Navigation

```
function constructCostmap():
    Odom ← getOdom();
    LaserScans ← getLIDAR();
    PointCloud ← getCamera();
    Costmap ← newCostmap(Costmap,LaserScans);
    Costmap ← newCostmap(Costmap,PointCloud);
return Costmap, Odom
function detectObstacles(Costmap, Position):
    for each OccupiedPosition in Costmap do
        Distance ← magnitude(OccupiedPosition -
          Position);
        if Distance < RobotRadius then
            return HasObstacles
        end if
    end for
return NoObstacles
function findStartPos(Costmap, Odom):
    Rays ← Costmap - Odom;
    StartPositionList ← [];
    for each RayVec in Rays do
        Detect ← detectObstacles(Costmap,RayVec);
        if Detect == NoObstacles then
            RandNum ← randomUniform(0, 1);
            Position ← RandNum*RayVec;
            StartPositionList.append(Position);
        end if
    end for
return StartPositionList
function createPath(StartPositionList, Costmap,
  VisitedPositionList):
    BestPath ← initPath();
    BestPathScore ← 0;
    for each StartPos in StartPositionList do
        for each EndPos in Costmap do
            Direction ← EndPos - StartPos;
            Num ← Int(magnitude(Direction)/0.5);
            Path ← arange(0,1,Num)*Direction +
              StartPos;
            Detect ← detectObstacles(Costmap,Path);
            if Detect == HasObstacles then
                continue;
            end if
            PathScore ←
              computeScore(Path,VisitedPositionList);
            if PathScore > BestPathScore then
                BestPathScore ← PathScore;
                BestPath ← Path;
            end if
        end for
    end for
return BestPath
function moveAlongPath(BestPath, Odom,
  Costmap, VisitedPositionList):
    for each Position in BestPath do
        Detect ← detectObstacles(Costmap,Position);
        if Detect == HasObstacles then
            moveRobotToPosition(Odom);
            break;
        end if
        moveRobotToPosition(Position);
        sleep();
        VisitedPositionList.append(Position);
    end for
return VisitedPositionList
```

between the start position and end position. Each position in the candidate path has a minimum distance of 0.5 m to the other positions. If a candidate path has an obstacle in its way, then it is rejected. A path score is computed for each candidate path, depending on the length of the path and the number of overlaps between the candidate path and the previously visited positions. When the candidate path goes through many of the previously visited positions, its path score is decreased. However, if the candidate path goes to unvisited locations, then its path score is increased. Longer paths also increase the path score. The candidate path with the highest path score is selected as the best path.

E. MOVING ALONG THE PLANNED PATH

As depicted in the function moveAlongPath, ROS2 Navigation 2 is used to move the Turtlebot3 along the best path. Navigation 2 controls the velocity and angular velocity in order to visit all the evenly spaced points in the best path. Before moving to a new location, the algorithm checks to see if there is an obstacle in the way. Upon detecting a blockage in its path, Turtlebot3 retraces its steps back to the start position (Odom). When the Turtlebot3 arrives at one of the planned positions, the current position is appended to the list of previously visited positions (VisitedPositionList). Subsequently, the robot waits at that location for a few minutes, while the ESP32C3 routers collect Wi-Fi and BLE packets from the Turtlebot3. Moreover, the ESP32C3 routers record the Unix times and positions of the robot for the training and testing databases. The process above repeats until all free positions on the map are sampled by the Turtlebot3.

IV. PACKET PROCESSING

A. ESP32C3 ROUTER

In order to capture packets, we built a custom router based on the ESP32C3 chip-set because it supports monitor mode on Wi-Fi and BLE simultaneously. It is able to determine RSSI from all Wi-Fi/BLE packets and supports Wi-Fi FTM to get the round trip time (RTT). For some BLE packets, the ESP32C3 provides the TX power of the mobile devices. An RF front end is added to increase the RX power and the

dynamic range of the received packets, while a SD card is added to store the data.

At the start of the day, the routers' real-time clocks (RTCs) are synchronized via simple network time protocol (SNTP) to the master server. The non FTM packets are timestamped by the RTCs with an accuracy of 1 μs. However, the FTM packets have a timestamp accuracy of 1 ns provided by the ESP32C3's high-resolution timer. Subsequently, raw packet data is written live to the SD card, and it is sent back to the master server at the end of the day. Most of the packet processing is done at the master server to reduce CPU load on the ESP32C3.

B. INCREASING WI-FI RESPONSE RATE USING THE ESP32C3

Manufacturers limit the power consumption of the Wi-Fi chipsets on the mobile devices to conserve battery charge. As a result, the number of packets transmitted by the mobile devices is small, and the localization accuracy is low. However, we can increase the localization accuracy by sending packets to the mobile devices using the ESP32C3 and getting a higher response rate. In order to cover most Wi-Fi channels, the routers alternate between channels 1, 6, and 11. After switching to a channel, the ESP32C3 transmits a request-to-send (RTS) packet to a mobile device. In response, the mobile device transmits a clear-to-send (CTS) packet back to the router, which contains information pertaining to the mobile device. The ESP32C3 can also send NULL packets to get an ACK response from the mobile devices. Moreover, special mobile devices can respond to Wi-Fi FTM requests, of which greatly increases the localization accuracy.

C. INCREASING BLE RESPONSE RATE USING THE ESP32C3

Similar to the Wi-Fi case, the ESP32C3 alternates between BLE channels 37, 38, and 39. Upon switching to a channel, the router sends a BLE scan request to a mobile device. Afterwards, the mobile device replies with a BLE scan response, of which contains BLE capabilities and sometimes model specific information. If the mobile device advertises a BLE service, then the ESP32C3 will send a pairing request packet to the device. Even if the pairing request is denied, the router will still receive a pairing response packet and the position of the mobile device.

D. SORTING WI-FI PACKETS BY SOURCE TYPE

Once the packets from the routers are collected, they are sorted and processed. The routers record the packets in non-chronological order. Therefore, the packet processor sorts the packets by timestamp in order to synchronize the packets from multiple routers. Note that malformed Wi-Fi packets are discarded due to having incorrect information. Afterwards, the packets are sorted by the type of wireless device: AP, wireless distribution system (WDS), bridged device, or mobile device. APs are found by looking at the source MAC addresses of the beacon frames. WDS are identified when the packets have ToDS=1 and FromDS=1. Bridged devices are identified when the packets have FromDS=1 and the source MAC addresses do not equal the BSSIDs. The remaining wireless devices are categorized as mobile devices. For the purposes of this paper, only the packets from the mobile devices are used for contact tracing.

E. SORTING BLE PACKETS BY SOURCE TYPE

Same as the Wi-Fi packet sorting and processing, the BLE packets are sorted by their arrival time. Subsequently, BLE packets with invalid cyclic redundancy check are discarded due to having incorrect protocol data unit types and incorrect manufacturer specific information. Afterwards, BLE packets are sorted by the type of TX address: public MAC addresses and random MAC addresses. Public BLE MAC address are stable and constant for long periods of time, so they are easily tracked and localized. On the other hand, random BLE MAC addresses rapidly change from one packet to another packet, and they require special BLE MAC de-randomization algorithms for tracking.

F. DEFEATING WI-FI MAC ADDRESS RANDOMIZATION

Many mobile devices randomize their Wi-Fi MAC addresses to prevent user tracking and identification [39]. In order to defeat Wi-Fi MAC address randomization, we create an algorithm to categorize mobile devices using model specific information from the probe requests. Firstly, we capture probe request frames emitted by the mobile devices. Every mobile device regularly transmits probe request packets, so this is not a problem. Secondly, we extract model specific information from the probe requests. Each device model type has unique model specific information such as supported rates, extended supported rates, high throughput (HT) capabilities, direct sequence (DS) parameter set, and vendor specific organizationally unique identifier (OUI). Furthermore, those model specific information are fixed and do not change over the lifetime of the device [40].

Thirdly, the model specific information is converted into a binary fingerprint vector. For example, if transmit beamforming is supported on the device, then it is set to "1" in the binary fingerprint vector, otherwise it is set to "0". In order to make the binary fingerprint vectors the same length, missing values are padded with "0". Fourthly, we use the binary hamming distance to compare binary fingerprint vectors. Even though two packets might have completely different MAC addresses, if the hamming distance of two packets' binary fingerprint vectors is zero, then the two packets originated from the same model type. This allows us to track and locate individual model types by collecting packets with the same binary fingerprint vectors. Finally, we use a ball tree clustering algorithm to categorize binary fingerprint vectors into their respective device types. The ball tree is constructed such that each leaf node contains the exact same binary fingerprint vector. Moreover, each branch contains device types from the same device family. As a

result, we can cluster unknown and new device types around well known device types.

G. EXTRACTING FEATURES FROM WI-FI PACKETS FOR LOCALIZATION

Specific Wi-Fi packet features are extracted as fingerprints for localization. Wi-Fi RSSI and Wi-Fi Signal Quality Index (SQI) correlate to the received (RX) powers of the routers. High RSSI values indicate the mobile devices are close to the routers and the RX power of the routers is large. Low RSSI values indicate mobile devices are far away from the routers and the RX power of the routers is small. Some wireless interfaces report the noise power of specific channels and packets. SQI is a function of RSSI and noise power. If the RSSI values are high, then the SQI values are high. Moreover, if the noise power is high, then the SQI values are low.

RSSI and SQI are susceptible to the type of mobile device, transmit (TX) power, and noise power. This makes RSSI and SQI somewhat unstable and dependent on the environmental conditions. On the other hand, some mobile devices support Wi-Fi FTM, of which greatly increases the robustness. To initiate FTMs, the router sends an FTM packet to the mobile device containing the FTM packet's departure time. Afterwards, the mobile device sends an ACK packet to the router containing the FTM packet's arrival time and the ACK packet's departure time. The router records the ACK packet's arrival time and computes the RTT. In order to obtain a more precise RTT, multiple series of FTM exchanges are performed and are averaged. ToF can be computed from RTT, and ToF is invariant to the type of mobile device, TX power, and noise power. In conclusion, ToF is far less susceptible to the external environment when compared to RSSI and SQI.

H. EXTRACTING FEATURES FROM BLE PACKETS FOR LOCALIZATION

The main disadvantage of Wi-Fi packets is the lack of TX power information. Every mobile device has a different TX power, and it results in different RX powers. Inconsistent RX powers produce incorrect localization predictions and invalid contact tracing paths. This is an open problem. We propose to use BLE packets to assist the Wi-Fi RX power calibration. Note that BLE packets contain the information of both TX powers and RX powers. The BLE path loss L_{BLE}

$$L_{BLE}(\vec{X}) = P_{BLETX} - P_{BLERX}(\vec{X}) \tag{1}$$

is computed using the constant BLE TX power P_{BLETX} and the BLE RX power P_{BLERX} as a function of position $\vec{X}$. It is invariant to the type of mobile device, since the path loss only depends on the distance to the router and the channel environment. Thus, the BLE path loss L_{BLE} is used as one of the inputs to the neural networks. Assuming the BLE path loss is equal to the 2.4 GHz Wi-Fi path loss at the same position $L_{BLE}(\vec{X}) = L_{WiFi}(\vec{X})$, the Wi-Fi TX power P_{WiFiTX}

$$P_{WiFiTX} = L_{BLE}(\vec{X}) + P_{WiFiRX}(\vec{X}) \tag{2}$$

can be calculated. Since the Wi-Fi TX power P_{WiFiTX} is constant for a specific model of mobile device, the Wi-Fi path loss L_{WiFi}

$$L_{WiFi}(\vec{X}) = P_{WiFiTX} - P_{WiFiRX}(\vec{X}) \tag{3}$$

can be computed. As a result, localization predictions using path loss have a higher accuracy.

I. NEURAL NETWORKS FOR LOCALIZATION

After extracting the features from the packets, they are fed into the BiLSTM neural networks to predict the positions of mobile devices. However, the BiLSTM neural networks require a large number of continuous trajectories for training. The training trajectories are generated via a simple recursive algorithm from the site survey sample locations. Firstly, a random position is selected as the current position $\vec{P}_i$. Secondly, another random position is selected as the candidate position $\vec{P}_C$ for the next position $\vec{P}_{i+1}$. If the Euclidean distance between the candidate position and the current position is less than a distance threshold $|\vec{P}_C - \vec{P}_i| < |k|$, then the candidate position becomes the next position $\vec{P}_{i+1} \leftarrow \vec{P}_C$; otherwise a new random position is selected as the candidate position $\vec{P}_C$. The distance threshold k is a random normal number that has a standard deviation of 1 m. Finally, the process above repeats until a trajectory of positions $\{\vec{P}_0, \vec{P}_1, \vec{P}_2, \ldots, \vec{P}_N\}$ is completed. For each floor in the building, 20,000 trajectories are randomly generated for training. Moreover, each trajectory has 20 different positions.

Once the training dataset is generated, it is used to train the BiLSTM neural networks. The mentioned neural networks have special neurons because they can retain information such as the past positions and the past features. This allows the BiLSTM to predict future positions using past positions and past features. Moreover, the inverse is also true because the BiLSTM can use future positions and future features to predict past positions. If ToF or SQI features are available, then they are fed into the network as a time series of features. If TX power is present, then signal path loss is used as an input feature to the network. When the features listed above are not available, the neural network defaults to RSSI for predicting the trajectories of mobile devices.

As shown in Fig. 1, the BiLSTM neural network consists of 2 BiLSTM layers followed by 2 dense layers. The number of input features to the network is denoted by F_{input} as it changes depending on the number of routers. Each BiLSTM layer consists of $7F_{input}$ neurons with tanh activation functions. The first dense layer has $14F_{input}$ neurons with LeakyReLU activation functions, while the second dense layer has 2 neurons with no activation function. Furthermore, the second dense layer outputs the predicted positions of a mobile device.

V. CONTACT TRACING ALGORITHM

The contact tracing algorithm takes a mobile device's MAC address/model name, an initial time and/or an initial position of a confirmed case as input and allows us to precisely track

Algorithm 2: Contact Tracing

```
function lookupConfirmedCase(Database,
 InitialTime, InitialPos, MACaddr, ModelName):
    UserInfos ← getValues(data=Database,
     key=InitialTime);
    if MACaddr != NULL then
        UserInfos ← getValues(data=UserInfos,
         key=MACaddr);
    else
        UserInfos ← getValues(data=UserInfos,
         key=ModelName);
    end if
    MinDistList ← [];
    for each User in UserInfos do
        AbsPos ← square(IntialPos - User.Pos);
        Dist ← sum(AbsPos, 2);
        Index ← minIndex(Dist);
        MinDistList.append(Dist[Index]);
    end for
    Index ← minIndex(MinDistList);
return UserInfos[Index]
function pathIntersect(UserInfo1, UserInfo2,
 DistanceThres, TimePeriodThres):
    DoesIntersect ← False;
    TimeMatrix ← transpose(UserInfo1.Times) -
     UserInfo2.Times;
    TimeMatrix ← abs(TimeMatrix);
    Index ← select(TimeMatrix < TimePeriodThres);
    PosMatrix ← transpose(UserInfo1.Pos[Index]) -
     UserInfo2.Pos[Index];
    PosMatrix ← square(PosMatrix);
    PosMatrix ← sqrt( sum(PosMatrix, 3) );
    Index ← select(PosMatrix < DistanceThres);
    if Index.size > 0 then
        DoesIntersect ← True;
    end if
return DoesIntersect
function findContacts(InputUsers,
 UserInfoList, AdjMat):
    UserSize ← UserInfoList.size;
    OutputUsers ← [];
    for i ← 1 To UserSize do
        TargetUser ← UserInfoList[i];
        if !InputUsers.contains(TargetUser) then
            continue;
        end if
        for j ← 1 To UserSize do
            CurrentUser ← UserInfoList[j];
            Result ← pathIntersect(TargetUser,
             CurrentUser);
            AdjMat[i][j] ← Result;
            if Result then
                OutputUsers.append(CurrentUser);
            end if
        end for
    end for
return AdjMat, OutputUsers
function createGraph(ConfirmedCases,
 UserInfoList, SearchDepth):
    UserSize ← UserInfoList.size;
    AdjMat ← zeroMatrix(UserSize, UserSize);
    InputUsers ← ConfirmedCases;
    for Depth ← 1 To SearchDepth do
        AdjMat, OutputUsers ←
         findContacts(InputUsers, UserInfoList,
         AdjMat);
        InputUsers ← OutputUsers;
    end for
return AdjMat
```

the paths of confirmed and suspected cases with an accuracy of 1.0 m. Furthermore, we can track droplet and surface exposures due to knowing the absolute positions of the users. The contact tracing procedure described in Algorithm 2 consists of 4 main parts: the key-value database system, looking up the paths of the confirmed cases, finding the suspected cases using path intersection, and creating a graph connecting the confirmed cases to the suspected cases.

A. KEY-VALUE DATABASE SYSTEM

For simplicity, a key-value database system is used to store the user information for the contact tracing system. Given a unique key, the algorithm can use it to look up a specific value in the database. Searching for values by comparing each key individually is extremely slow because it takes $O(N)$ time. However, hashmaps can decrease the search time to $O(1)$ constant time. Our system is built on Redis, a hash based key-value database system, where it hashes the unique key to obtain a pointer. Afterwards, the pointer is used to access the memory location of the associated value. In particular, Redis uses the cyclic redundancy check (CRC) hash function family to lookup key-value pairs because it is simple and has hardware acceleration in modern CPUs. As a result, Redis speeds up the user information look-ups in the contact tracing system.

The contact tracing database contains many key-value pairs. Each value contains the user's device model name, MAC addresses, positions, date/time, medical test results, and contacts with other users. Furthermore, an initial position or a MAC address can be used as a key to look up those values. These properties are useful for looking up the paths of the confirmed cases.

B. LOOKING UP THE PATHS OF THE CONFIRMED CASES

The first step of the contact tracing algorithm is to lookup user information of the confirmed cases. However, many users have the exact same identifiers such as the same trajectory, the same device model name, and the same random MAC addresses. As a result, significant ambiguity is present in the lookup process. To solve the problem above, we create the lookupConfirmedCase function to reduce the identifier

ambiguity. There are 4 main scenarios, where the function has to perform look-ups:

Scenario A: all mobile devices do not have MAC address randomization. This causes each mobile device to have a single unique MAC address that is easily identified and tracked by the algorithm. When a patient has a positive test result, they only need to provide their single MAC address (MACaddr) and initial time (InitialTime) to look-up user information (UserInfos). The initial time requires the specific day and hour of the arrival in the location.

Scenario B: all mobile devices have MAC address randomization, but each mobile device has a unique model name. Due to the fact that each unique model name emits a unique probe request signature, we can still identify and track individual mobile devices. This time, the patient has to provide the device model name (ModelName) and the initial time. For example, the device model name could be iPhone 13, Galaxy S22, or Pixel 6. As a result, the algorithm can look up the information on the confirmed case without knowing the actual MAC addresses.

Scenario C: all mobile devices have MAC address randomization and multiple devices have the exact same model name. However, devices with the same model names have unique trajectories that do not have intersecting points with each other. This scenario is far more difficult than Scenario B, and the algorithm can only tell users apart by their unique trajectories. Thus, the patient needs to provide the initial position (InitialPos), model name, and initial time. The input initial position could be any position on the patient's trajectory. The algorithm selects the user information that contains the closest trajectory to the initial position.

Scenario D: all mobile devices have MAC address randomization and multiple devices have the exact same model name. Furthermore, multiple devices with the same model names have intersecting trajectories with each other or near misses. In some situations, two physically separated trajectories might be mislabelled as having an intersecting point because the neural networks predicted the wrong positions. Subsequently, the algorithm falsely groups multiple individual users as a single confirmed case. This increases the false positive rate and adds more users to the list of suspected cases. On the other hand, the false negative rate stays the same because the algorithm still traces the correct social contacts.

C. FINDING THE SUSPECTED CASES USING PATH INTERSECTION

After retrieving the user information of the confirmed cases, we use the pathIntersect function to find the suspected cases. Given the paths of User1 and User2, the function determines if their paths intersect within a certain distance and time threshold. Firstly, we compute the time differences (TimeMatrix) between both user paths. Secondly, we select specific positions (PosMatrix) from the user paths that have time differences less than the time period threshold (TimePeriodThres). Typically, public health authorities will input a different time period threshold for each type of pathogen. Thirdly, we select positions from the user paths that are closer than the distance threshold (DistanceThres). We set the distance threshold equal to 2.0 m because our localization accuracy is around 1.0 m. If there exists at least one distance less than the distance threshold, then the function indicates an intersection, otherwise the function does not indicate an intersection.

D. CREATING A GRAPH CONNECTING THE CONFIRMED CASES TO THE SUSPECTED CASES

Using the path intersection function, we create a graph connecting the confirmed cases to the suspected cases. A graph is defined as a set of nodes that are connected by edges. The adjacency matrix $AdjMat$ describes every edge connection, where $AdjMat[i][j] = True$ represents a connection between node i and node j. On the other hand, $AdjMat[i][j] = False$ represents no connection between node i and node j. In particular, each user is assigned to a unique node and each social contact is represented by an edge connection. To begin, the function createGraph retrieves the total number of users (UserSize), and the adjacency matrix is initialized as a zero matrix of size UserSize by UserSize. Afterwards, the list of confirmed cases (ConfirmedCases) is selected as the list of input users (InputUsers). The findContacts function compares the input users' paths to every other user path in the list of total users. If their paths intersect, then the corresponding element in the adjacency matrix is updated $AdjMat[i][j] = True$ and the new social contact is appended to the output list (OutputUsers). Pathogens can spread very rapidly due to infecting their primary contacts and later the secondary contacts of those primary contacts. The contact tracing algorithm gets ahead of the disease spread by recursively applying the findContacts function until the search depth (SearchDepth) is reached. This produces a social contact graph that is very deep and has many degrees of separation. In conclusion, the createGraph function returns the fully built adjacency matrix connecting the confirmed cases to the suspected cases.

E. USER PRIVACY CONSIDERATIONS

Privacy is a very important aspect to keeping collected information safe and within regulations with Canadian and British Columbia Privacy Acts. In our system, phone numbers, email addresses, and legal names are not collected by the routers and are not stored in the database. Only the MAC addresses of the Wi-Fi/BLE chipsets and device model names are obtained as the identification of the mobile devices. Note that MAC addresses cannot directly identify users, and MAC address randomization also complicates the mapping of multiple MAC addresses to user devices. Public health authorities will only map the MAC addresses to user identities for confirmed cases and suspected contact cases, with the help of additional information of user identity and device wireless interface MAC addresses. Furthermore, as part of the privacy protection, users that enter a building

with the contact tracing system in place need to be aware of what the system does and actively consent to their data being collected. Users also have the ability to retroactively erase their information in the contact tracing database. Finally, the collected data such as MAC addresses, device model name, and positions are encrypted with AES256 algorithm.

VI. RESULTS AND DISCUSSION

Multiple datasets were collected at the University of Victoria, Victoria, British Columbia, Canada in the Engineering Office Wing (EOW) 3rd floor, EOW 4th floor, Engineering Computer Science (ECS) 1st floor, and ECS 5th floor. At each floor, the Turtlebot3 physically moves along 3 unique trajectories. Every trajectory contains unique positions that the other trajectories do not have. One of the trajectories is randomly selected for the training dataset, while another is selected for the testing dataset. The remaining trajectory is appended to the cross-validation dataset. Furthermore, we artificially generated more training trajectories using the data points from the training dataset as described in Section II Subsection D. However, we did not create artificial trajectories using the testing and cross-validation datasets. The quality of the data collected by the ESP32C3 routers is unknown, thus we use commercial off the shelf (COTS) routers as a reference to validate the quality of the data from the ESP32C3 routers. The data collected are organized into two main groups: Dataset A is collected using the COTS routers and Dataset B is collected using the ESP32C3 routers.

A. DATASET INFORMATION

Dataset A contains the packets collected by COTS routers. At every position, at least 50 samples are obtained by the routers. Each sample contains a timestamp, X position, Y position, θ orientation, Wi-Fi RSSI, Wi-Fi SQI, BLE RSSI, and BLE TX power. For the EOW 3rd floor, 11 Wi-Fi routers and 7 BLE routers are deployed to obtain the dataset. Note that some Wi-Fi routers share the same locations as the BLE routers. The raw dataset contains approximately 500,000 samples at 1,000 different positions, where each sample has 31 wireless parameter features. For the EOW 4th floor, 9 Wi-Fi routers and 7 BLE routers are deployed to obtain the dataset. There are fewer routers on this floor due to the lack of power outlets. The raw dataset contains approximately 300,000 samples at 600 different positions, where each sample has 29 wireless parameter features. For the ECS 1st floor, 7 Wi-Fi routers and 7 BLE routers are deployed to obtain the dataset. The raw dataset contains approximately 200,000 samples at 1000 different positions, where each sample has 24 wireless parameter features. For the ECS 5th floor, 8 Wi-Fi routers and 6 BLE routers are deployed to obtain the dataset. The raw dataset contains approximately 300,000 samples at 600 different positions, where each sample has 24 wireless parameter features extracted from the packets.

Dataset B is sampled at the same locations and with the same procedures as Dataset A. However, Dataset B uses ESP32C3 routers, and it provides a new wireless feature known as Wi-Fi FTM. Moreover, the ESP32C3 routers occupy 40 MHz bandwidth instead of the 20 MHz bandwidth in Dataset A. These new additions increase the localization accuracy of the BiLSTM and the precision of the contact tracing algorithm.

B. IDENTIFYING UNIQUE MOBILE DEVICES FROM RANDOM MAC ADDRESSES

In this section, the effectiveness of the clustering algorithm for identifying unique mobile devices from random MAC addresses is tested. For the test setup, MAC address randomization is enabled on the devices, and they are forced to join a wireless network. Every time a mobile device joins a new wireless network, the operating system generates a new random MAC address for that specific network. Ground truth MAC addresses are obtained by looking at the settings menu. Simultaneously, the devices' probe request packets are captured at the router side. Afterwards, the clustering algorithm is applied to the probe requests to identify unique mobile devices from random MAC addresses.

Table 2 shows the results of the clustering algorithm on the testing dataset. Each row of the table contains a single bucket, of which each bucket contains MAC addresses that belong to the same mobile device. Galaxy S4 is loaded with LineageOS 16, of which does not have MAC address randomization. Table 2 shows the clustering algorithm assigning Galaxy S4's single MAC address to a single bucket and MAC addresses from other devices are not present in that bucket. The result matches the Galaxy S4's ground truth MAC address. Similar to the Galaxy S4, the HTC One X does not have MAC randomization, and it results in a single MAC address found in Table 2. However, Galaxy S6 is loaded with LineageOS 18.1, and it generates a new random MAC address upon joining a new wireless network. Galaxy S6 is forced to join 7 different wireless networks, and the clustering algorithm places all 7 of the Galaxy S6's random MAC addresses in the same bucket. Note that the clustering algorithm has 100% accuracy because all the MAC addresses in Galaxy S6's bucket in Table 2 matches all the MAC addresses in the ground truth. On the other hand, Android 11 on Galaxy A11 adds a new feature that randomizes MAC addresses while scanning for nearby SSIDs. The exact same test is performed on the Galaxy A11, of which the ground truth MAC addresses matches the clustering result found in Table 2. Note that the extra MAC addresses of Galaxy A11 are generated when scanning for SSIDs. We have also observed that Android only generates a new random MAC address on the first network connection. Rejoining a previously connected network yields the same MAC address.

The iPhone SE supports MAC address randomization because it has iOS 15.1 firmware. For the test, iPhone SE is forced to join 7 different wireless networks, and the clustering algorithm places all 7 of the iPhone SE's random MAC addresses in the same bucket. Again, the clustering algorithm achieves 100% accuracy because all the ground truth MAC addresses are found at the iPhone SE's bucket in Table 2.

TABLE 2: Test Results of the Clustering Algorithm for Identifying Unique Mobile Devices from Random MAC Addresses.

Bucket	Device	MAC Addresses
0	Galaxy S4	D0:22:BE:F5:7C:B4
1	HTC One X	E8:99:C4:99:57:24
2	Galaxy S6	4E:0F:A0:57:F8:75, 26:45:19:1E:D5:FE, 1A:5B:0A:B1:7D:4A, 0E:BF:6D:4D:ED:A7, 42:B2:3B:14:49:F9, 1A:CF:16:13:A2:CB, 8C:F5:A3:3D:16:DA, 3A:DC:D3:0A:46:B6
3	Galaxy A11	6A:E0:23:0C:20:0F, 56:5C:AC:D6:13:30, E2:01:19:D0:64:2D, FA:05:BB:EA:47:2D, A0:27:B6:EE:6A:A7, 7E:69:90:C6:C4:04, DA:00:FD:35:82:25, 56:2F:2B:64:BC:C5, F6:08:C4:AF:61:94, 16:0D:FA:80:F8:1F, 5E:99:98:7B:5A:BF, 96:96:27:97:22:4C, FE:CB:1A:2E:F5:9A, B2:78:9D:5C:B9:1A 16:3C:FC:DF:1C:CA, 96:38:7C:5D:20:5C
4	iPhone SE	82:31:01:8A:F3:AD, AE:9E:BE:7A:F3:D3, A6:E9:93:A7:9D:3E, D2:C5:A7:8B:9E:2C, 46:33:10:CE:43:3B, AA:CB:57:97:5E:5F, 1A:40:6D:01:B4:05, 96:C2:5B:09:D8:4E, 3A:E6:E3:9B:8E:6D, 56:D3:41:61:0B:0A, A2:68:13:44:B2:EF, 8E:6A:CF:EF:6E:1F, 56:A2:4A:EE:D4:46, E2:F7:83:DC:1E:E4, 22:29:5A:0D:F3:24, B6:33:3F:4F:89:1A, 9E:3D:78:F4:38:5D, ...
5	iPhone X	86:98:6E:73:89:1D, 86:AD:C7:47:02:39, 3E:6F:2D:B3:4D:BB, 76:8A:CB:74:73:90, 9A:2D:E5:A8:F1:5A, 76:34:D2:C0:89:71, FE:B8:15:02:43:7C, 76:55:81:98:C3:78, CE:56:BC:E7:3E:72, 46:C9:78:16:41:B6, BE:F2:DB:37:1A:8A, 2A:2F:3E:B1:C7:A0, 6E:4C:1E:F1:8E:E8, A2:97:F2:BA:2A:D5, 6A:DD:55:59:2E:68, DA:D2:D1:55:18:60, F2:C5:62:AD:29:04, ...
6	PinePhone	7A:26:59:B4:C6:6D, D6:8A:05:6A:62:F5, 96:61:D9:88:25:45, 7E:53:9C:5F:BE:D0, B6:13:70:3F:28:C9, 0A:70:BB:F2:2D:9D, 52:A2:3B:BD:6D:DF, D6:DB:E0:37:8C:CE, 62:0B:F8:3C:3A:E2, BA:12:FB:78:53:F4, A2:3E:F7:DF:14:03, BE:B6:47:61:BC:31, EA:14:84:75:F9:00, 8E:B7:F1:4D:1A:FC, C6:41:5C:E2:C6:7B, 92:47:59:89:C4:37, CA:DC:C0:CF:39:FB, ...

The extra MAC addresses found in the iPhone SE's bucket are generated when scanning for nearby SSIDs. The iPhone X's results are the same as the iPhone SE's results because they both have the same iOS 15.1 firmware.

The full Arch Linux distribution is installed onto the PinePhone, of which allows full control over MAC address randomization. We wrote a script to generate 25 new random MAC addresses and to save them into a file as the ground truth. Afterwards, the clustering algorithm's results found in Table 2 are compared to the ground truth. The clustering algorithm is able to place all the PinePhone's MAC addresses into the same bucket without any other MAC addresses from other devices being there. Overall, the clustering algorithm correctly classified every single test device into their respective buckets, even though their MAC addresses are randomized. However, any two mobile devices that have the same model number at the same spacetime might cause the system to produce incorrect results. This is due to identical devices producing indistinguishable probe request information and RSSI information.

C. PATH ANALYSIS OF DATASET A

The BiLSTM neural networks are applied to many environments, and their RMSE and MAE performances are shown in Table 3. At EOW 3rd floor, the BiLSTM with Wi-Fi has a RMSE of 0.83 m and a MAE of 0.58 m. Moreover, only using BLE information yields a similar RMSE of 0.88 m and a MAE of 0.56 m because the Wi-Fi routers share the same positions as the BLE routers. Combining Wi-Fi and BLE information together results in a RMSE of 0.82 m and a MAE of 0.58 m, of which has no discernible difference. For error analysis, the ground truth trajectory is compared against the BiLSTM's predicted trajectory in Fig. 2a. The path begins at (X = 0 m, Y = 0 m) with medium error of 1.0 m. However, the error increases to 1.5 m at the corners because that position has the least amount of LoS from the routers. Subsequently, the highest error of 2.5 m occurs in the east hallway because there are multiple objects blocking the signal paths of the routers. Afterwards, the error rapidly drops to 0.5 m as the BiLSTM recovers itself and gets back on the correct trajectory. For the rest of the path, the error predominantly stays below 1.0 m, but there are a few locations where the error jumps above 1.0 m due to corners.

At EOW 4th floor, the BiLSTM with Wi-Fi has a RMSE of 0.92 m and a MAE of 0.61 m. The RMSE of EOW 4th floor is slightly higher than the RMSE of EOW 3rd floor because the routers' signal paths in EOW 4th floor are blocked by more walls and doors. Moreover, the number of routers is reduced from 11 to 9 due to the lack of power outlets. Using only BLE produces a similar RMSE of 0.93 m and a MAE of 0.63 m due to the number of Wi-Fi and BLE routers being similar. Combining Wi-Fi and BLE information together results in a slightly lower RMSE of 0.84 m and a MAE of 0.60 m. The lower RMSE and MAE is caused by the increased bandwidth and the increased channel diversity. Just as before, the ground truth trajectory is compared to the BiLSTM's predicted trajectory in Fig. 2b. This time, the highest error of 2.3 m occurs at the starting position of (X = 0 m, Y = 0 m). The large error is caused by not having enough space and power outlets to place more routers at the starting position. Soon after, the error quickly decreases to 1.0 m as the BiLSTM recovers and gets back on the correct trajectory. There are a few instances where the error jumps significantly due to the objects blocking the router's signals, but those errors are lower than the starting position errors.

ECS 1st floor is very different from the other floors because the packets are collected in an open area instead of an enclosed hallway. In an open area, Wi-Fi RSSI and BLE RSSI changes approximately 5 dBm per 10.0 m. The routers are not sensitive enough to detect the small changes in RSSI and creates errors in localization. Moreover, there is an extra degree of freedom compared to the hallways, of which creates more ambiguity in the trajectory. As a result, localization errors on this floor are much larger than the other floors. For Wi-Fi, the RMSE is 1.69 m and the MAE is 1.42 m, of which the localization errors are significantly higher than EOW 3rd floor, EOW 4th floor, and ECS 5th

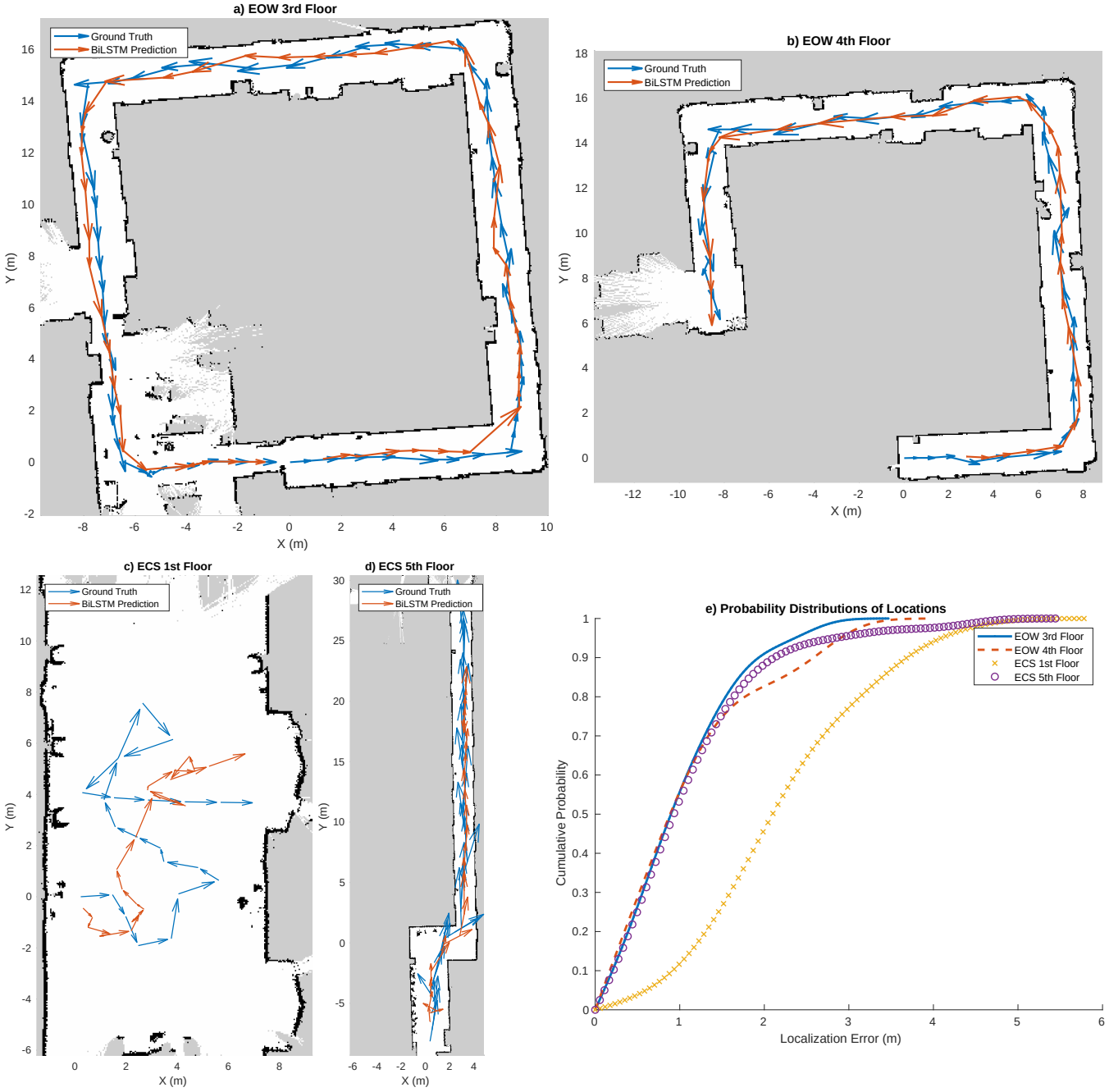

FIGURE 2: BiLSTM's predicted locations vs ground truth at: a) EOW 3rd floor b) EOW 4th floor c) ECS 1st floor d) ECS 5th floor e) probability distributions of Dataset A Wi-Fi+BLE

floor. Localization using BLE information has a larger RMSE of 2.13 m and a larger MAE of 1.73 m. This is caused by the BLE having a lower transmit power and worse antennas. Combining Wi-Fi and BLE slightly lowers the RMSE to 1.30 m and the MAE to 1.03 m because the antenna diversity and the channel diversity are increased. The ground truth trajectory is compared to the BiLSTM's predicted trajectory in Fig. 2c. The trajectory starts off well at the origin of (X = 0 m, Y = 0 m) with an error of 0.5 m. However, the error rapidly increases to above 2.0 m because the predicted trajectory quickly diverges from the ground truth. The BiLSTM never recovers from incorrect predictions, and the error remains above 2.0 m. The large errors are caused by the ambiguity of the wireless features. Multiple unique positions on the map have the same Wi-Fi RSSI, SQI, and BLE RSSI.

The ECS 5th floor has a similar layout to EOW 3rd floor and EOW 4th floor because they are all located in a hallway. As a consequence, the localization performance on ECS

TABLE 3: Dataset A: BiLSTM's localization performance at different locations.

Location	Method	APs	RMSE (m)	MAE (m)	Training Time (s)	Testing Time (μs)
EOW 3rd Floor	BiLSTM+Wi-Fi	11	0.83	0.58	3.67	420
EOW 3rd Floor	BiLSTM+BLE	7	0.88	0.56	3.12	399
EOW 3rd Floor	BiLSTM+Wi-Fi+BLE	18	0.82	0.58	4.34	445
EOW 4th Floor	BiLSTM+Wi-Fi	9	0.92	0.61	3.55	410
EOW 4th Floor	BiLSTM+BLE	7	0.93	0.63	3.02	402
EOW 4th Floor	BiLSTM+Wi-Fi+BLE	16	0.84	0.60	3.96	405
ECS 1st Floor	BiLSTM+Wi-Fi	7	1.69	1.42	4.72	340
ECS 1st Floor	BiLSTM+BLE	7	2.13	1.73	3.31	361
ECS 1st Floor	BiLSTM+Wi-Fi+BLE	14	1.30	1.03	4.32	417
ECS 5th Floor	BiLSTM+Wi-Fi	8	0.83	0.62	3.72	370
ECS 5th Floor	BiLSTM+BLE	6	0.87	0.68	3.38	373
ECS 5th Floor	BiLSTM+Wi-Fi+BLE	14	0.92	0.63	4.13	391

Note: Wi-Fi implies Wi-Fi RSSI and SQI,while BLE implies BLE RX power and TX power.

TABLE 4: Dataset B: BiLSTM's localization performance at different locations.

Location	Method	APs	RMSE (m)	MAE (m)	Training Time (s)	Testing Time (μs)
EOW 3rd Floor	BiLSTM+Wi-Fi FTM	8	0.80	0.57	4.59	360
EOW 3rd Floor	BiLSTM+Wi-Fi RSSI	8	0.82	0.62	5.57	366
EOW 3rd Floor	BiLSTM+Wi-Fi FTM+Wi-Fi RSSI	16	0.75	0.55	5.93	442
EOW 5th Floor	BiLSTM+Wi-Fi FTM	8	0.75	0.57	4.21	461
EOW 5th Floor	BiLSTM+Wi-Fi RSSI	8	0.77	0.59	3.70	428
EOW 5th Floor	BiLSTM+Wi-Fi FTM+Wi-Fi RSSI	16	0.70	0.52	3.52	373
ECS 1st Floor	BiLSTM+Wi-Fi FTM	8	1.52	1.31	11.63	426
ECS 1st Floor	BiLSTM+Wi-Fi RSSI	8	1.63	1.41	11.89	409
ECS 1st Floor	BiLSTM+Wi-Fi FTM+Wi-Fi RSSI	16	0.89	0.70	12.07	446

Note: Wi-Fi RSSI implies Wi-Fi RSSI and SQI, while Wi-Fi FTM implies RTT using IEEE 802.11mc .

5th floor is very similar to the other floors. This is further evidenced by the BiLSTM with Wi-Fi on ECS 5th floor having an RMSE of 0.83 m and a MAE of 0.62 m, which is comparable to the RMSE of 0.83 m and the MAE of 0.58 m on EOW 3rd floor. Moreover, the BiLSTM with BLE on ECS 5th floor has a RMSE of 0.87 m and a MAE of 0.68 m that is similar to the RMSE of 0.88 m and the MAE of 0.56 m on EOW 3rd floor. The ground truth trajectory is compared to the BiLSTM's predicted trajectory in Fig. 2d. The predicted path has low RMSE at the origin as it is surrounded by many routers. Moreover, the localization RMSE is stable when the trajectory moves upwards. However, the RMSE jump increases drastically to 2.0 m at the end of the path due to the signal reflections at the corner of the hallway.

The cumulative distribution function (CDF) of the localization errors is shown in Fig. 2e, and it tells the same story as above. The EOW 3rd floor, EOW 4th floor, and ECS 5th floor have very similar CDFs with expected localization errors of approximately 0.89 m. Again, this is due to the floors having similar map layouts. On the other hand, ECS 1st floor is an outlier with significantly different CDF. Most of the errors in ECS 1st floor occur between 1.0 m and 3.0 m, raising the expected localization error to 2.3 m. The significant increase in error is caused by the ECS 1st floor having more degrees of freedom, more possible trajectories, and ambiguity in the wireless features.

D. PATH ANALYSIS OF DATASET B

Introducing Wi-Fi FTM to the Dataset B generally improves localization accuracy due to having more independent wireless features. For EOW 3rd floor, Table 4 shows the BiLSTM with Wi-Fi RSSI has a RMSE of 0.82 m and a MAE of 0.62 m. This is very similar to BiLSTM and Wi-Fi RSSI in Dataset A. However, using Wi-Fi FTM yields a RMSE of 0.80 m and a MAE of 0.57 m. Combining Wi-Fi RSSI amd Wi-Fi FTM together results in a RMSE of 0.75 m and a MAE of 0.55 m, of which have slightly lower errors than Wi-Fi and BLE in Dataset A. For error analysis, the ground truth trajectory is compared against the BiLSTM's predicted trajectory in Fig. 3a. The path begins at (X = 0 m, Y = 0 m) with low error of 0.5 m. As the trajectory moves in the counterclockwise direction, the error stays below 1.0 m. However, the error increases to above 1.5 m at specific corners because the routers' LoS are broken. Moreover, the error also increases to 1.5 m at the end of the trajectory due to signal scattering in the room.

In the open area of ECS 1st floor, the BiLSTM and Wi-Fi RSSI yields a RMSE of 1.63 m and a MAE of 1.41 m. The localization error is very large and is comparable to the BiLSTM and Wi-Fi in Dataset A. Localization using Wi-Fi FTM has a RMSE of 1.52 m and a MAE of 1.31 m, of which has little change in error. Combining Wi-Fi RSSI and Wi-Fi FTM drastically lowers the RMSE to 0.89 m and the MAE to 0.70 m because having more independent wireless features decreases the position ambiguity in localization. Moreover,

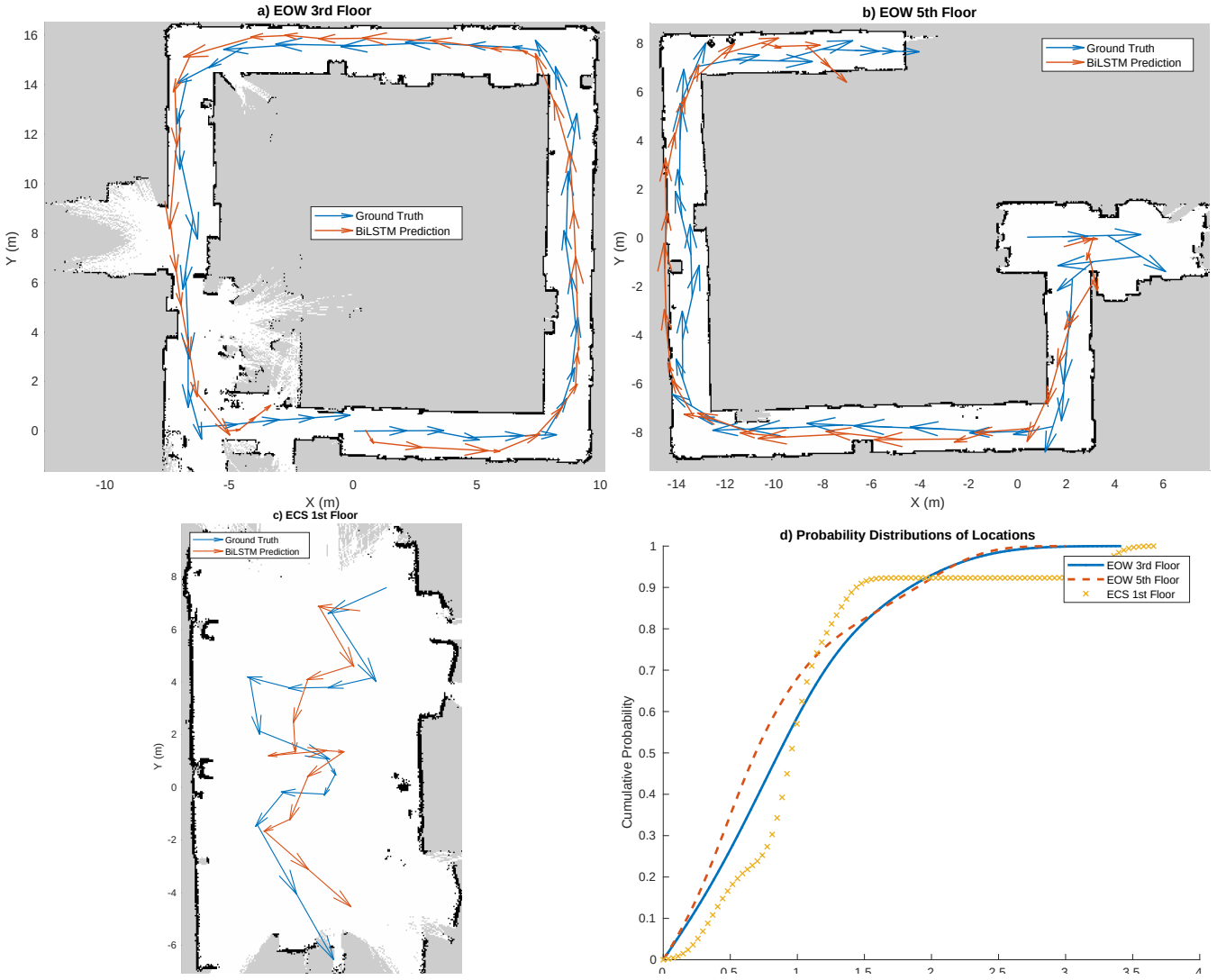

FIGURE 3: BiLSTM's predicted locations vs ground truth at: a) EOW 3rd floor b) EOW 4th floor c) ECS 1st floor d) probability distributions of Dataset B Wi-Fi+FTM

having more features increases redundancy in case one of the features is corrupted by the channel noise. The ground truth trajectory is compared to the BiLSTM's predicted trajectory in Fig. 3c. The trajectory at the origin starts a high error of 1.3 m. On the first turn, the error drops to 0.5 m. Moreover, the error hovers around 1.0 m when moving in a straight line. However, the error increases to 2.0 m at the second turn due to BiLSTM failing to predict sharp turns. Afterwards, the BiLSTM recovers and the error drops below 1.0 m for the rest of the path.

E. CONTACT TRACING WEBSITE

We developed a contact tracing website that allows the public health authorities to find the suspected cases using the details of the confirmed cases. Fig. 4 shows an example of the website, where the authorities enter the initial date, time, and position of the confirmed case. Afterwards, the authorities enter the MAC address/model name of the specific mobile device. The pathogen's lifetime is inputted as the search time period. A longer time period increases the search window and finds more social contacts. Subsequently, the search graph depth controls the traversal depth of the social contact graph. A graph depth of 1, lists suspected cases that have direct contact with the confirmed case. On the other hand, a graph depth of 3, lists suspected cases within 3 social contacts of the confirmed case. Moreover, the graph depth provides indirect contacts together with direct contacts.

For simplicity, the confirmed case is marked in red, while the suspected cases are marked in green, blue, orange, purple, and pink. On the top right, we plot the x and y positions of the suspected and confirmed cases. The bottom right lists the suspected cases along with the model name, MAC addresses,

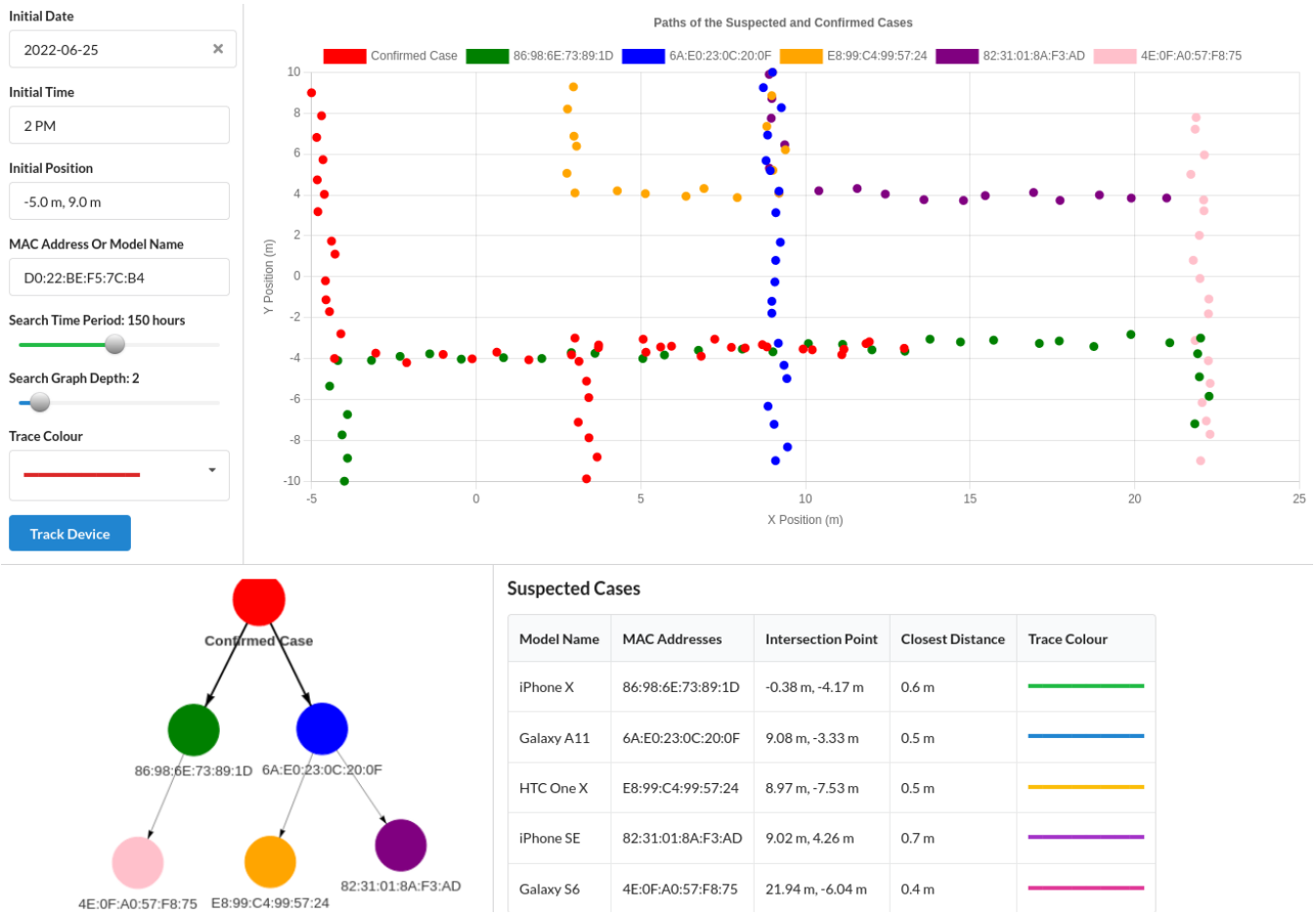

FIGURE 4: The contact tracing website allows users to find the suspected cases using the details of the confirmed cases.

intersection point, and closest distance. The red path intersects with the green path, and this is indicated in the first row of the suspected cases list, where it shows the intersection point of (-0.38 m, -4.17 m) and the closest distance of 0.6 m. Furthermore, a social contact graph is displayed in the bottom left, where it shows a link between the red circle and the green circle.

VII. CONCLUSION

We have created a novel, privacy preserving Wi-Fi and BLE contact tracing system for finding the detailed paths of the infected individuals without any user intervention. The system tracks smartphones, but it does not require smartphone applications, connecting to the routers, or any other extraneous devices on the users. A custom built autonomous Turtlebot3 is used for site survey simulating user movement and smartphone transmission. The smartphones' received power, transmit power, and round trip time are collected by custom ESP32C3 routers. Even though MAC randomization is employed in modern smartphones, we have defeated it to track many devices. Afterwards, the wireless parameters collected are converted to signal path loss and ToF, of which the BiLSTM takes and predicts the absolute paths of the users. The localization performance and the RMSE is always below 0.9 m for Wi-Fi RSSI + Wi-Fi FTM. Public health authorities can use the designed website to find the paths of the confirmed cases and suspected cases, together with their MAC addresses/smartphone model specific information. They can also track indirect contact transmissions originating from surfaces and droplets.

REFERENCES

[1] D. Majra, J. Benson, J. Pitts, and J. Stebbing, "SARS-CoV-2 (COVID-19) superspreader events," Journal of Infection, 2020.
[2] K. T. Eames and M. J. Keeling, "Contact tracing and disease control," Proceedings of the Royal Society of London. Series B: Biological Sciences, vol. 270, no. 1533, pp. 2565–2571, 2003.
[3] R. A. Kleinman and C. Merkel, "Digital contact tracing for COVID-19," CMAJ, vol. 192, no. 24, pp. E653–E656, 2020.
[4] S. Altmann, L. Milsom, H. Zillessen, R. Blasone, F. Gerdon, R. Bach, F. Kreuter, D. Nosenzo, S. Toussaert, J. Abeler et al., "Acceptability of app-based contact tracing for COVID-19: Cross-country survey study," JMIR mHealth and uHealth, vol. 8, no. 8, p. e19857, 2020.
[5] R. Hinch, W. Probert, A. Nurtay, M. Kendall, C. Wymant, M. Hall, K. Lythgoe, A. B. Cruz, L. Zhao, A. Stewart et al., "Effective configurations of a digital contact tracing app: a report to NHSX," Retrieved July, vol. 23, p. 2020, 2020.
[6] N. Nanthini, B. M. Shankar, S. S. Kumar, and R. S. Ganesh, "Deep learning approach for minimizing disease spread using face identification and contact tracing," 2020 Fourth International Conference on I-SMAC

(IoT in Social, Mobile, Analytics and Cloud)(I-SMAC), pp. 527–532, 2020.

[7] M. Yaghi, T. Basmaji, R. Salim, J. Yousaf, H. Zia, and M. Ghazal, "Real-time Contact Tracing During a Pandemic using Multi-camera Video Object Tracking," 2020 International Conference on Decision Aid Sciences and Application (DASA), pp. 872–876, 2020.

[8] J. Gardy, J. Johnston, S. Sui, V. Cook, L. Shah, and et al., "Whole-genome sequencing and social-network analysis of a tuberculosis outbreak," New England Journal of Medicine, vol. 364, no. 8, pp. 730–739, 2011.

[9] Hallam Stevens and Monamie Bhadra Haines, "TraceTogether: Pandemic Response, Democracy, and Technology," East Asian Science, Technology and Society: An International Journal, vol. 14, no. 3, pp. 523–532, 2020. [Online]. Available: https://doi.org/10.1215/18752160-8698301

[10] T. Sharon, "Blind-sided by privacy? Digital contact tracing, the Apple/Google API and big tech's newfound role as global health policy makers," Ethics and Information Technology, vol. 23, no. 1, pp. 45–57, 2021.

[11] S. Ahmed, Y. Xiao, T. T. Chung, C. Fung, M. Yung, and D. D. Yao, "Privacy guarantees of Bluetooth Low Energy contact tracing: A case study on COVIDWISE," Computer, vol. 55, no. 2, pp. 54–62, 2022.

[12] K. Woodward, E. Kanjo, D. O. Anderez, A. Anwar, T. Johnson, and J. Hunt, "DigitalPPE: low cost wearable that acts as a social distancingreminder and contact tracer," Proceedings of the 18th Conference on Embedded Networked Sensor Systems, pp. 758–759, 2020.

[13] T. Shelby, T. Caruthers, O. Y. Kanner, R. Schneider, D. Lipnickas, L. E. Grau, R. Manohar, L. Niccolai et al., "Pilot evaluations of two Bluetooth contact tracing approaches on a university campus: Mixed methods study," JMIR Formative Research, vol. 5, no. 10, p. e31086, 2021.

[14] L. Reichert, S. Brack, and B. Scheuermann, "A survey of automatic contact tracing approaches using Bluetooth Low Energy," ACM Transactions on Computing for Healthcare, vol. 2, no. 2, pp. 1–33, 2021.

[15] Q. Zhao, H. Wen, Z. Lin, D. Xuan, and N. Shroff, "On the accuracy of measured proximity of Bluetooth-based contact tracing apps," International Conference on Security and Privacy in Communication Systems, pp. 49–60, 2020.

[16] D. J. Leith and S. Farrell, "Coronavirus contact tracing: Evaluating the potential of using bluetooth received signal strength for proximity detection," ACM SIGCOMM Computer Communication Review, vol. 50, no. 4, pp. 66–74, 2020.

[17] P. Di Marco, P. Park, M. Pratesi, and F. Santucci, "A Bluetooth-based architecture for contact tracing in healthcare facilities," Journal of Sensor and Actuator Networks, vol. 10, no. 1, p. 2, 2021.

[18] E. Hernández-Orallo, C. T. Calafate, J.-C. Cano, and P. Manzoni, "Evaluating the effectiveness of COVID-19 Bluetooth-based smartphone contact tracing applications," Applied Sciences, vol. 10, no. 20, p. 7113, 2020.

[19] P. G. Madoery, R. Detke, L. Blanco, S. Comerci, J. Fraire, A. G. Montoro, J. C. Bellassai, G. Britos, S. Ojeda, and J. M. Finochietto, "Feature selection for proximity estimation in COVID-19 contact tracing apps based on Bluetooth Low Energy (BLE)," Pervasive and Mobile Computing, vol. 77, p. 101474, 2021.

[20] M. Cunche, A. Boutet, C. Castelluccia, C. Lauradoux, and V. Roca, "On using Bluetooth-Low-Energy for contact tracing," Inria Grenoble Rhône-Alpes; INSA de Lyon, 2020.

[21] H. Gorji, M. Arnoldini, D. F. Jenny, A. Duc, W.-D. Hardt, and P. Jenny, "STeCC: Smart Testing with Contact Counting Enhances COVID-19 Mitigation by Bluetooth app based contact tracing," medRxiv, 2020.

[22] T. M. Yasaka, B. M. Lehrich, and R. Sahyouni, "Peer-to-peer contact tracing: development of a privacy-preserving smartphone app," JMIR mHealth and uHealth, vol. 8, no. 4, p. e18936, 2020.

[23] I. Nakamoto, S. Wang, Y. Guo, and W. Zhuang, "A QR code–based contact tracing framework for sustainable containment of COVID-19: Evaluation of an approach to assist the return to normal activity," JMIR mHealth and uHealth, vol. 8, no. 9, p. e22321, 2020.

[24] A. S. Hoffman, B. Jacobs, B. van Gastel, H. Schraffenberger, T. Sharon, and B. Pas, "Towards a seamful ethics of Covid-19 contact tracing apps?" Ethics and Information Technology, pp. 1–11, 2020.

[25] D. Mobo, A. L. R. Garcia et al., "Using automated contact tracing system app with QR code to monitor and safeguard parishioners against COVID-19 at St. Anthony of Padua Parish, Subic, Zambales," American Research Journal of Computer Science and Information Technology, vol. 4, no. 1, pp. 1–4, 2020.

[26] S. Wang, S. Ding, L. Xiong et al., "A new system for surveillance and digital contact tracing for COVID-19: spatiotemporal reporting over network and GPS," JMIR mHealth and uHealth, vol. 8, no. 6, p. e19457, 2020.

[27] A. Trivedi, C. Zakaria, R. Balan, A. Becker, G. Corey, and P. Shenoy, "WiFiTrace: Network-based contact tracing for infectious diseases Using passive WiFi sensing," Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies, vol. 5, no. 1, pp. 1–26, 2021.

[28] G. Li, S. Hu, S. Zhong, W. L. Tsui, and S.-H. G. Chan, "vContact: Private WiFi-based IoT contact tracing with virus lifespan," IEEE Internet of Things Journal, 2021.

[29] K. Fallahi, C.-T. Cheng, and M. Fattouche, "Robust Positioning Systems in the Presence of Outliers Under Weak GPS Signal Conditions," IEEE Systems Journal, vol. 6, no. 3, pp. 401–413, 2012.

[30] V. Gokhale, G. M. Barrera, and R. V. Prasad, "FEEL: Fast, Energy-Efficient Localization for Autonomous Indoor Vehicles," arXiv preprint arXiv:2102.00702, 2021.

[31] J. Tiemann and C. Wietfeld, "Scalable and precise multi-UAV indoor navigation using TDOA-based UWB localization," 2017 international conference on indoor positioning and indoor navigation (IPIN), pp. 1–7, 2017.

[32] V. Moreno, M. A. Zamora, and A. F. Skarmeta, "A low-cost indoor localization system for energy sustainability in smart buildings," IEEE Sensors Journal, vol. 16, no. 9, pp. 3246–3262, 2016.

[33] H. Obeidat, W. Shuaieb, O. Obeidat, and R. Abd-Alhameed, "A review of indoor localization techniques and wireless technologies," Wireless Personal Communications, pp. 1–39, 2021.

[34] M. T. Hoang, B. Yuen, X. Dong, T. Lu, R. Westendorp, and K. Reddy, "Recurrent neural networks for accurate RSSI indoor localization," IEEE Internet of Things Journal, vol. 6, no. 6, pp. 10 639–10 651, 2019.

[35] M. T. Hoang, Y. Zhu, B. Yuen, T. Reese, X. Dong, T. Lu, R. Westendorp, and M. Xie, "A soft range limited k-nearest neighbors algorithm for indoor localization enhancement," IEEE Sensors Journal, vol. 18, no. 24, pp. 10 208–10 216, 2018.

[36] B. Kane, C. A. Zajchowski, T. R. Allen, G. McLeod, and N. H. Allen, "Is it safer at the beach? Spatial and temporal analyses of beachgoer behaviors during the COVID-19 pandemic," Ocean and Coastal Management, vol. 205, p. 105533, 2021.

[37] R. Amsters and P. Slaets, "Turtlebot 3 as a robotics education platform," International Conference on Robotics in Education (RiE), pp. 170–181, 2019.

[38] E. Erős, M. Dahl, K. Bengtsson, A. Hanna, and P. Falkman, "A ROS2 based communication architecture for control in collaborative and intelligent automation systems," Procedia Manufacturing, vol. 38, pp. 349–357, 2019.

[39] J. Martin, T. Mayberry, C. Donahue, L. Foppe, L. Brown, C. Riggins, E. C. Rye, and D. Brown, "A Study of MAC address randomization in mobile devices and when it fails," Proceedings on Privacy Enhancing Technologies, vol. 4, pp. 268–286, 2017.

[40] Q. Robyns, Bonne and Lamotte, "Noncooperative 802.11 MAC Layer fingerprinting and tracking of mobile devices," Hindawi Security and Communication Networks, vol. 5, pp. 1–27, 2017.