The following publication T. Kulshrestha, D. Saxena, R. Niyogi and J. Cao, "Real-Time Crowd Monitoring Using Seamless Indoor-Outdoor Localization," in IEEE Transactions on Mobile Computing, vol. 19, no. 3, pp. 664-679, 1 March 2020 is available at https://dx.doi.org/10.1109/TMC.2019.2897561.

Real-time Crowd Monitoring using Seamless Indoor-Outdoor Localization

Abstract— Human identification and monitoring are critical in many applications, such as surveillance, evacuation planning. Human identification and monitoring are not an easy task in the case of a large and densely populated crowd. However, none of the existing solutions consider seamless localization, identification, and tracking of the crowd for surveillance in both indoor and outdoor environments with significant accuracy. In this paper, we propose a novel and real-time surveillance system (named, SmartISS) which identifies, tracks and monitors individuals' wireless equipment(s) using their MAC ids. Our trackers/sensing units (PSUs) are the portable entities comprising of Smartphone/Jetson-TK1/PC which are enough to capture users' devices probe requests and locations without users' active cooperation. *PSUs* upload collected traces on the cloud server periodically where cloud server keeps finding the suspicious person(s). To retrieve the updated information, we propose an algorithm (named, LLTR) to select the optimal number of *PSUs* for finding the latest location(s) of the suspicious person(s). To validate and to show the usability of *SmartISS*, we develop a real prototype testbed and evaluate it extensively on a real-world dataset of 117,121 traces collected during the technical festival held at IIT Roorkee, India. *SmartISS* selects *PSUs* with an average selection accuracy of 95.3%.

Index Terms— Surveillance system, Localization, MAC, Wi-Fi, Trajectory analysis, Outlier/Anomaly detection, Smartphone

1 INTRODUCTION

RECENT advances in mobile devices with a wide range of communication and low-cost location sensing techniques have emerged as a new era of ubiquitous computing. Location-aware information processing module is a kind of ubiquitous computing in which the user's location can be effectively used for disaster/crisis management, public safety, and evacuation path planning and supportive tasks during emergency situations. In public safety, timeliness of information generation, processing, and dissemination are the most significant factors for coordination, evaluation, analysis, future prevention, and strategies [1]. Moreover, the real-time assistance creates an environment of trust and credibility among people and mitigates the critical situations effectively.

As the movement of crowd is highly unpredictable [2], [3], surveillance systems allow administrative authorities to monitor the crowd and their movement simultaneously from the remote locations. During the case of an emergency, such as stampede, fire, suspicious activity, etc., these systems can provide the immediate and efficient corrective actions. Moreover, emergency situations need to be detected at an initial point to alleviate the risk of a situation evolving towards a dangerous incident.

Human identification, tracking, and monitoring are not easy tasks in the case of a large and densely populated crowd. Visual sensor, like camera is used to track places of a person in the crowd while some research works are emphasizing on wireless technology/single positioning, such as, Wi-Fi enabled devices [4], Smartphone's GPS/General Packet Radio Service (GPRS) [5], RFID [6], and Bluetooth (BT)/Bluetooth Low Energy (BLE) tags [7] to accurately recognize and track humans in indoor and outdoor scenarios.

Regardless of the current advances in the computer vision and pattern recognition techniques, it is a challenging task to get the global condition of the crowd (identification, tracking, and monitoring) from the video footage during gatherings [8], [9], such as congregations, rallies, etc. Existing crowd monitoring systems [4], [6], [10], [11] based on the above-mentioned techniques work well for individuals and not for mass gatherings as a whole. Moreover, these techniques are not accurate for finding the location of individuals moving in indoor and outdoor environments seamlessly. However, single wireless technology-based localization approach is not robust for tracking individuals seamlessly in indoor and outdoor environments due to their trade-off among power consumption, accuracy, and coverage area as well as single point of failure [12].

To handle the above-mentioned issues, to the best of our knowledge, we are the first to develop an interactive and intelligent real-time Smartphone-based surveillance system (named, SmartISS) for public safety. *SmartISS* uses the Smartphone/Jetson TK1 [13]/PC based sensing units to identify, monitor, and track individuals' equipment(s) (Smartphones/BT/BLE) using their MAC ids (non-participatory) and a hybrid localization technique, GPS-Wi-Fi-Cellular (Google location API [14]). Sensing units upload collected individuals' traces on the cloud server periodically where cloud server keeps finding the suspicious person(s) using an outlier detection algorithm. *SmartISS* allows administrative authorities to find the latest location(s) of that suspicious person(s) in the real-time. To find the latest locations/trajectory of a suspicious person only from the cloud server is not sufficient as it can have outdated data while querying all sensing units deployed in dispersed locations will increase the communication cost. To minimize the response time and communication cost, selection of the appropriate sensing unit(s) should be optimal. Therefore, we propose an algorithm to select the optimal number of sensing units deployed at geographically dispersed locations to retrieve required information with high sensing unit(s) selection accuracy and low response time. Retrieving the information from the large database(s) in real-time is also a challenging task regarding computation and access time. Therefore, *SmartISS* also allows top-*k* query on multi-dimensional data. Top-*k* query retrieves *k* data objects from the selected sensing units. As a proof-of-concept, we develop a prototype of the system and evaluate it extensively using a real-world dataset.

In summary, the main technical contributions of this paper are as follows.

1. We propose a novel real-time intelligent surveillance system (named, SmartISS) to find the suspicious person(s) and his/her latest location(s)/trajectory seamlessly in both indoor and outdoor environments.
2. We propose an algorithm (named, LLTR) to select the optimal number of sensing units deployed at geographically dispersed locations for finding the latest location(s) of the suspicious person(s) with high sensing unit (s) selection accuracy and low response time. *SmartISS* also allows top-*k* query on multi-dimensional data.
3. Extensive experiments on real prototype testbed having a real-world dataset of more than 117,121 traces collected during the technical festival of 3 days, Cognizance 2017 in IIT Roorkee, India show the usability and validity of *SmartISS* system. The *SmartISS* selects the sensing units to generate the trajectory of the suspicious person(s) with an average selection accuracy of 95.3 percent.

The rest of the paper is organized as follows. In Section 2, we discuss the related works. Section 3 covers the background of our proposed system. Section 4 and 5 explain the system model and system architecture, respectively. In Section 6, we elaborate prototype implementation details while Section 7 describes about experimental evaluation. Finally, Section 8 concludes the paper.

2 Literature Survey

In this section, we shall discuss the related works of *SmartISS*.

Wirz, et al. [15] developed a Smartphone-based crowd monitoring system through participatory sensing. The system identifies the crowd density based on the individuals walking speed. Koshak, et al. [16] explained how to leverage GPS and Geographic Information Systems (GIS) technologies for tracking the pedestrian movement through computer software during the haj pilgrimage. Authors have analyzed flow rates and levels of services at different places and instances in Makkah. Based on this information, volunteers can easily find out the critical areas and the time at which pilgrims should take precautions.

Musa, et al. [17] proposed a trajectory tracking system for tracking unmodified smart devices passively and to use access points hardware for information retrieval. Rose, et al. [18] presented the Argos sensor network-based approach for exploring the usage and details of wireless access points for tracking Wi-Fi equipped transport vehicles, and the user's wireless devices for behavior analysis through the emitted probe requests.

Cheng, et al. [19] explored the previous researches based on spatio-temporal based users' behavior, users' physical closeness, and device association history. Hong, et al. [20] proposed a system to analyze the interaction patterns and social behavior of users carrying Smartphones through monitoring the Wi-Fi probes and null data frames emitted by Smartphones. Barbera, et al. [21] performed the social-network analysis and sociological aspects of users, such as language, vendor adoption, etc., from the collected probe requests of wireless devices during the national and international events. Bonné, et al. [22] implemented a low-cost Raspberry Pi based system to capture and process the wireless signals for monitoring and analyzing human mobility using real-life movement data in the university campus and the music festival. Luzio, et al. [23] developed a system which analyzes the device owner's social belonging information from the probe requests collected from the wireless devices. Wang, et al. [24] presented an approach to monitor the Smartphone's Wi-Fi signals in real time for analyzing the human queues w.r.to the service/waiting time, and human flow, etc. This approach can be useful in many applications, such as dynamic workflow scheduling, shift assignments, etc.

In addition, some authors also used RFID, BT, etc., for identifying and tracking individuals in different environments. Bahl, et al. [25] presented a radio-frequency (RF) based user locating and tracking system (RADAR) placed in indoor environments. Versichele, et al. [26] used a proximity-based Bluetooth tracking scheme to identify the spatio-temporal data of visitors at Ghent University festivities events. Furthermore, authors have also explored the statistics, such as flow maps of individuals attending events, individuals' counts, and share of returning individuals. Oh, et al. [27] proposed an intelligent surveillance system in which it gathers heterogeneous sensory information (visual and RFID sensors), creates/updates database information for tracking and identifying objects. The system assigns a global object number to each identified object for maintaining consistent information for the same object.

Single wireless technology cannot accurately track an individual who is passing through indoor and outdoor locations seamlessly. It is well known that GPS could not give high accuracy in the indoor, urban and dense area due to the multipath issues [28]. While, the use of Wi-Fi for outdoor environments do not work well in compared to GPS. On the other hand, cellular-based localization schemes perform well iff more base stations are deployed near to the localization area. Furthermore, RFID-based tracking and monitoring systems use the expensive RFID readers and have human body interference and low coverage area related issues in the densely crowded places [29]. BT is also a possible solution for indoor localization [30] due to its low-cost and low-power consumption. But, BT has a low range for scanning and high discovery time to track the human at large crowd. While, BLE tags based on RSSI measurements suffer from huge location errors due to non-uniform shadowing [31]. To overcome the limitations of the RSSI, researchers proposed to use Channel State Information (CSI) [32]–[34] for highly accurate and reliable indoor localization. Implementation of CSI in a smart device for location tracking requires to have a NIC card whose driver has been hacked to expose the CSI information to the respective application. Moreover, these techniques are not robust because of their trade-off among accuracy, power consumption, and coverage area as well as a single point of failure.

Experimental research to use the different sensing units, such as Smartphone, Jetson TK1, PC for identifying, tracking, and monitoring the individuals in the mass gatherings for surveillance is in the nascent state. Most of the surveillance systems are based on the video footage and other sensing techniques, like RFID, BT, BLE, etc., which require either a rich set of hardware or does not work for a large crowd as a whole and makes the system costly and complex. Moreover, developers have only provided piecemeal solutions to the crowd identification, tracking, and monitoring system without considering the overall implementation of a complete end-to-end system.

3 BACKGROUND

We first introduce probe requests and then, we discuss how sensing units capture the probe requests and intercept the MAC ids.

A sensing unit (Smartphone/Jetson TK1/PC) is used as a portable wireless scanner for capturing and processing nearby wireless frames emitted by smart devices, such as Smartphones, BLE/BT tag, NFC equipped tags/devices. To find an access point (AP) to get associated, a smart/wireless device keeps sending probe requests, i.e., 802.11 Management frames [35] periodically. A smart device keeps looking for better AP, even if it is already connected to an AP. In our case, smart devices acting as sensing units with an external Wi-Fi adapter and working in monitor mode (named, Portable Sensing Unit (PSU)) captures and intercepts these probe requests passively [36]. Monitor mode means a sensing unit can capture all probe requests in its scanning range sent over the wireless medium, even if those probe requests are not destined for it. Moreover, it is not required for a sensing unit (i.e., PSU) to get connected with a user device (non-participatory).

A frame header of 802.11 Management frame comprises of the following main fields: *Frame Type*, *Subtype*, *Transmitter Address*, *Receiver Address*, etc., where the Transmitter Address (TA) is a MAC address of the user's smart device. *TA* remains in the plain text (not encrypted) even if a device is connected to a secure network. All captured probe requests remain intact, i.e., without removing any frame header. Whenever a new frame is received, an *OnFrameReceived* event handler is called to process the received frame for finding the Automated Gain Control (AGC) and *TA* (if exists, as some frames may not contain *TA*). In addition, it is also possible to find that a received frame is of an *AP* or user's smart device through the *FromDS* bit in the frame control field (the B9 bit). In our analysis, we did not use an *AP's* MAC address as it is not related to any person.

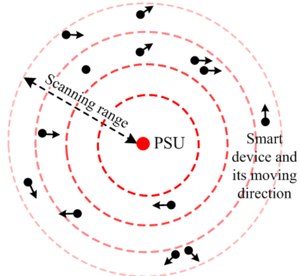


Fig. 1. *PSU* scanning individuals' devices present within its sensing range

Fig. 1 shows a setup of a *PSU* sensing individuals' devices. There is no need of any extra hardware at *PSU* for sensing probe requests and Internet access on the individuals' devices of the persons being tracked; neither tracking requires any application installed on the individual's devices. If two or more *PSUs* are present in each other's transmission range, they can share their data with each other through Wi-Fi direct. A *PSU* can use the Internet connection of another *PSU* available in its proximity for data uploading.

4 SYSTEM MODEL AND PROBLEM STATEMENT

4.1 System Model

We design and develop an intelligent surveillance system which identifies, tracks, and monitors crowd for finding the suspicious person(s) and their recent locations. *SmartISS* comprises of three main components: a set of *PSUs* $P = \{PSU_1, PSU_2, \ldots, PSU_n\}$, a cloud server ($C_{Server}$) and an application server (A_{Server}). These components are connected/communicated through Wi-Fi/3G/4G.

The system model works as follows: *SmartISS* at the sensing side is the combination of software and hardware which supports the identification, tracking, and monitoring of the smart devices enabled individuals available in the vicinity of sensing units. For achieving high location accuracy, *PSU* uses the hybrid GPS-Wi-Fi-Cellular-based positioning technique for tracking individuals seamlessly in indoor and outdoor environments. *PSUs* capture the frames transmitted on wireless media by smart devices from the persons in the crowd, extract the MAC ids, store them, and send locally processed and filtered data to C_{Server} for further long-term processing.

SmartISS system utilizes the sensed and collected data for anomaly detection, trajectory analysis, and emergency services through C_{Server}. The C_{Server} uses the *k-d* tree and an efficient, in-memory *k*-nearest neighbor algorithm for detecting the anomalies/outliers [37], [38], [39]. The C_{Server} passes MAC address and the last detected location of that outlier to A_{Server} for finding the outlier's latest location(s) from the C_{Server} and *PSUs* deployed at different regions.

The C_{Server} can have outdated data while, querying all

PSUs is expensive w.r.to time and the number of messages. It is a big challenge to get an accurate list of *PSUs* having the information of requested MAC address. Therefore, we propose an algorithm to select the optimal number of *PSUs* deployed at different regions for efficiently retrieving latest location(s) of an outlier. Moreover, *SmartISS* can perform top-k query on multi-dimensional localization data at C_{Server} and *PSUs*. In A_{Server}, all query messages are handled by Extensible Messaging and Presence Protocol (XMPP) framework [40].

4.2 Problem Statement

Problem definition

With the aforementioned system model, the formal definition of the proposed system *SmartISS* can be given as follows:

Definition 1. *Given n mobile/static sensing units deployed randomly in the task area and m MAC addresses of individuals' devices captured by sensing units with detected time and location, SmartISS finds the suspicious person(s) and selects the optimal number of sensing units n' (where n' ≤ n) deployed at geographically dispersed locations such that the following objectives are achieved:*

1) *Retrieve latest location(s) of suspicious person(s) in real-time with high PSU(s) selection accuracy and low response time.*
2) *Retrieve top-k relevant data objects from multi-dimensional data in low response time.*

Definition 2. *Optimal PSUs selection problem: Given a set of PSUs S_u and location of the suspicious person, select a subset S of S_u such that accuracy of S is maximized over all possible subsets in less response time.*

To sense m individuals, we have a total number of n *PSUs*. Each individual will be tracked atleast by a *PSU* deployed randomly in the task area. So, there will be a collection of k sets (i.e., $1 \leq k \leq 2^n$) of *PSUs*, where each set will have the tracking information of some individuals IN_i where $1 \leq i \leq m$.

Finding the optimal solution to the *PSU* selection problem is NP-hard. We prove the NP-hardness of the optimization problem by giving a polynomial-time reduction from the NP-hard set cover problem.

Theorem 1. *The optimal PSU selection algorithm is NP-hard.*

Proof: Recall that there are total 2^n sets of *PSUs* (n is finite), where each set will have the tracking information of atleast an individual, i.e., each individual corresponds to a set of S_u. Then, the decision version of the optimal *PSU* selection problem can be transformed into the following set cover problem: given the universal set of *PSUs* $S_u = \{u_1, u_2, \ldots, u_n\}$, and a collection C of 2^n sets whose union comprises the universe, determine whether there are subset $C' \subseteq C$, such that every element in S_u belongs to atleast one member in C'?

The decision version of the optimal *PSU* selection problem is equivalent to that of the set cover problem [41] which is NP-complete. Therefore, optimal *PSU* selection problem is NP-hard.

5 System Architecture

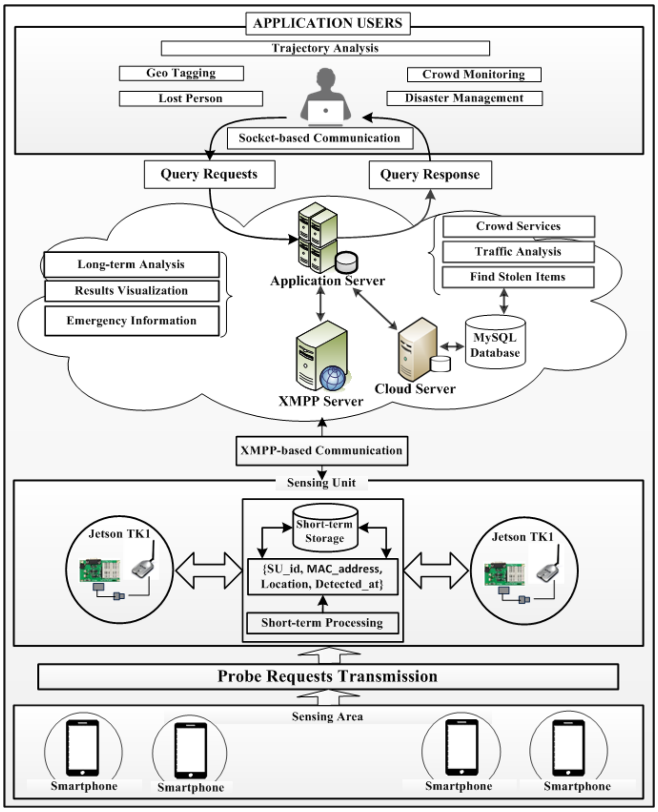


Fig. 2. *SmartISS* architecture

Fig. 2 shows the proposed *SmartISS* system architecture. *SmartISS* system works in a hierarchical fashion where the sensing units (static/dynamic) deployed at the lowest layer collects probe requests from the smart devices of the individuals. Then, sensing units intercept MAC addresses and store them for a period of time with the detected location and timestamp at which that probe request is captured. Whenever C_{Server} detects any outlier, it passes that information to the A_{Server} for finding the latest location(s) of the detected outlier. The functionality of the *SmartISS* system is divided into three main phases: 1) data collection and local data processing at sensing units (Processing@PSUs); 2) remote data processing at C_{Server} (Processing@$\mathrm{C_{Server}}$); and 3) data retrieval at A_{Server} (Processing@$\mathrm{A_{Server}}$). We shall discuss these phases sequentially in the next sections.

5.1 Processing@PSUs

Whenever an individual reaches the sensing range of any sensing unit deployed at geographically different locations, *PSU* (both dynamic and static) captures the probe requests emitted by individual's devices. Then, *PSU* extracts the MAC address from probe request and maintains an entry in its internal memory with the following attributes <*SU_id*, *MAC_address*, *Location*, *Detected_at*> where *SU_id* denotes the sensing unit's name, *MAC_address* is MAC id of individual's device, *Location* is the location of the *PSU* and *Detected_at* is time of individual's device detection. After uploading a file to C_{Server}, PSU deletes the corresponding file from its internal memory. If uploading fails, it retries to upload the file when the network connection is available again. The file can be uploaded using either Wi-Fi or mobile data as decided by the operating system. There is NO relation between $PSUs$ deployment and location accuracy. One PSU is enough for scanning an individual MAC address, appending other information and uploading that data to the C_{Server}. When a PSU intercept a MAC address from a probe request, PSU adds its own current location in that record.

As PSU adds its location at the users' location traces and PSU uses multiple wireless technologies, GPS-Wi-Fi-Cellular (Google Location API) for finding its current location, the granularity of the location accuracy depends on the Google Location API. As per the statistics [42] of the Google Location API, the maximum accuracy of the Smartphone Galaxy Nexus is 10 m, and 40 m approximately when the location update interval is set to 5 seconds (high accuracy) and 20 seconds (balanced power), respectively. Suppose, the location accuracy of a PSU is 10 m and the range of a PSU scanning is 30 m. Then we can say that in worst case, a user will be in the range (user's location accuracy) of 30 ± 10 m (*PSU_scanning_range* ± *PSU_location_accuracy*) which is well commensurate with the peak demand of a Smartphone-based surveillance system.

For security and privacy purpose, we apply Rivest Cipher 4 (RC4) [43] to the filtered MAC addresses. Encrypted MAC addresses are put into a temporary file along with timestamps of their detection. Only the authorized persons are able to access the real MAC address of any person's smart device. Furthermore, for handling the duplicate MAC addresses, all entries in the *PSUs* are maintained in the *HashMap*. Whenever a MAC address is added to the cache *HashMap*, it is passed to the MAC detection logger. It saves the MAC address, current location, and timestamp in a temporary file named *temp*. When the number of records in one file reaches above a certain threshold (e.g., 50), the temporary file is renamed to the current UNIX timestamp and the file is enqueued for upload. A new file name *temp* is created for storing new records. The records are saved in JSON format to upload without any modification. Instead of uploading a single record to the C_{Server}, we store few records in the internal storage of the application and then send it to the C_{Server} at once. It saves battery power and network bandwidth. However, the data at C_{Server} would be a little bit outdated.

Furthermore, in recent times a MAC address randomization technique has been proposed to secure Smartphones from being stalked as these phones pass through Wi-Fi environments. Smart devices use globally unique MAC address when a device is *connected* or *attempts* to connect to an *AP*. *SmartISS* handles the MAC randomization through the association/authentication frames [44] and discard the locally assigned MAC addresses. As we deploy *SmartISS* in the smart campus (all area is covered through Wi-Fi) IIT Roorkee, Smart devices remain associated or want to get associated with the *APs*, therefore, association/authentication frames-based approach is the best solution in our application scenario for dealing with MAC randomization in real-time. In addition, *SmartISS* can also easily opt for other MAC derandomization attacks, such as UUID-E reversal, device signature, Karma attack, and control frame attack [45], [46].

5.2 Processing@C_{Server}

Each location has a semantic significance associated with the activity. The presence of a student in a semantic location refers that the student is performing that activity. For instance, the library implies self-study, lecture hall implies attending classes, and canteen implies eating. Most of the time, there is a certain periodicity, frequency, and order in which a student visits these locations. This contains information that can be utilized to build powerful predictors of future behavior. Any change or deviation from this ordinary behavior is an *anomaly/outlier*.

$PSUs$ keep uploading collected data to C_{Server} at a pre-specified time interval, and then C_{Server} applies outlier detection algorithm on the uploaded data. The C_{Server} detects anomalies/outliers in the current location dataset using dynamic *k-d* tree (*k*-dimensional tree) with the nearest neighbor algorithm. A *K-d* tree is a space partitioning data structure (binary search tree) where data in each node is organized in a *k*-dimensional space. *K-d* trees are appropriate in range and nearest neighbor searches.

Given the location coordinates of individuals collected through *SmartISS*, we need to find out the location coordinates which are not close to the other points. We divide the whole area of IITR (1.48 km^2) into grid cell of 53 x 53 m where we have total 28 x 28 cells. Then, we put all collected records during a time interval in the grids. We iterate all grid cells to find out single traces in grid cells. Then, we find out neighbors of those single detected points using Euclidean distance (can be available in the neighbor grid cells) where we set Euclidean distance for outlier detection to 30 m. If no one exists in neighbor cells for those single detected points, then those points are referred as an *anomaly*. Furthermore, some rooms in each event locations were accessible to only organizing team members, and during the experiments, those areas are considered as *restricted areas*. If a new MAC address (apart from registered MAC addresses of security personals and authorities) is present in any restricted area, then that MAC address is also considered as an *anomaly*.

The anomaly information is published into the C_{Server}. Since the admin authorities are registered with C_{Server}; they can take efficient corrective actions immediately once a suspicious person is detected. After finding the outlier(s), C_{Server} passes the information <*MAC id*, *location coordinates*, and *timestamp*> to A_{Server} for getting the latest locations of a detected outlier. The format of the query is as follows: <*last_n'*, *attributes_list*, *MAC_address*> where *last_n'* is the number of recent locations of the outlier to retrieve, *attributes_list* contains a list of *n'*, *detected_at*, and *SU_id*, while *MAC_address* is MAC id of the outlier's device.

5.3 Processing@A_{Server}

Whenever an outlier (O) is detected at C_{Server}, it passes outlier's MAC address and the last detected location to A_{Server} where A_{Server} sends it to the XMPP server. E.g., *last_3&location&detected_at&SU_id&MAC_address*. A_{Server} deals with the XMPP server and C_{Server} while XMPP server deals with *PSUs* only. XMPP forwards the original message to selected *PSUs* list to get up-to-date information about outlier.

Optimal *PSU* selection algorithm is used to find the latest trajectory of the detected outlier(s). Each individual will have followed a specific route instead of visiting all deployed *PSUs*. To get the latest location(s) of outlier from *PSUs*, the selection of appropriate *PSU*(s) should be optimal. The primary challenge to query is how to choose the most accurate list of *PSUs.* If all *PSUs* are queried, it will increase the communication cost and time. Similarly, a query to a random number of *PSUs* does not ensure availability of requested data. To query only the C_{Server} is also not sufficient as C_{Server} can have outdated data. To handle the problems as mentioned above, it is required to design and develop a new intelligent approach to query the appropriate *PSUs* for low response time with high *PSU* selection accuracy. Therefore, we propose an algorithm to select the optimal number of *PSUs* deployed at different regions for retrieving latest locations of a particular MAC address (named, LLTR) efficiently.

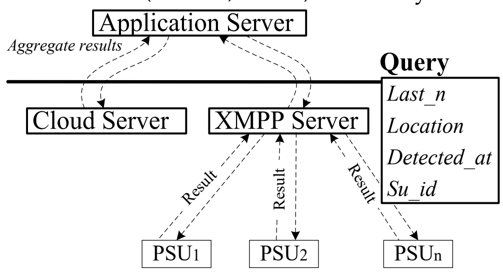


Fig. 3. A query and results propagating through the network

To select the appropriate list of *PSU*(s), we use the XMPP communication to make our *SmartISS* system fast and reliable. XMPP server uses the dynamic *k-d* tree. It then uses requested MAC address to search the *k*-nearest neighbors using the Nearest Neighbor (NN) approach. Now, XMPP server sends the top-*k* query to the selected *PSUs*. Selected *PSUs* compute the local top-*k* data objects and forward the response to XMPP server. When a response arrives at XMPP server from the list of selected *PSUs*, XMPP server decodes the message into dictionary *object* and returns a response to the A_{Server}.

When A_{Server} gets a response from XMPP server, it checks the response message to ensure that it has a result of all attributes mentioned in the query. If the response contains incomplete results, then A_{Server} parses the same query into SQL and sends it to the C_{Server}. After getting a response from the C_{Server}, A_{Server} merges this data with the previous incomplete response and sorts them (see Fig. 3). Then, sorted response based on *Detected_at* is provided to the administrative authorities. Furthermore, a *map matching module* projects each location onto a corresponding road segment where those points were truly generated and produces the trajectory on the basis of timestamps.

In the next sub-section, we shall discuss our proposed Latest Locations Retrieval (LLTR) algorithm.

Proposed LLTR Algorithm

In this section, we discuss the algorithm used for retrieval of latest locations of the queried outlier (see Fig. 4). In Table 1, we list the variables used to describe the pseudocode of *LLTR* algorithm.

TABLE 1
VARIABLES USED IN LLTR ALGORITHM

Variables	Significance
$MAC_{address}$	MAC id of detected anomaly/outlier *A*
L_{loc}	Last detected location of anomaly/outlier *A*
n'	# of requested locations
R_{PSU}	List of responses when L_{loc} == *null*
R_{ϕ}	# of empty responses
R'_{PSU}	List of responses when $L_{loc} \neq$ null
$R_{\tilde{\phi}}$	# of non-empty responses in R'_{PSU}
R''_{PSU}	List of responses when $L_{loc} \neq$ null and $R_{\tilde{\phi}} = = 1$
R'''_{PSU}	List of responses when $L_{loc} \neq$ null and $R_{\tilde{\phi}} > 1$
R'_{ϕ}	# of responses when $L_{loc} \neq$ null
R''_{ϕ}	# of responses when $L_{loc} \neq$ null and $R_{\tilde{\phi}} = = 1$
R'''_{ϕ}	# of responses when $L_{loc} \neq$ null and $R_{\tilde{\phi}} > 1$
All_{PSU}	List of all *PSUs*
K_{PSU}	Contains the list of *k*-nearest *PSUs*
C_{loc}	Current location of the *PSU* which provided the data
RL_{loc}	Recent location of the anomaly/outlier *A*

Whenever C_{Server} detects an outlier, it checks the outlier's timestamp, i.e., *detected_at*. If the difference between the *detected_at* and the current timestamp is less than a threshold, then the returned location from the C_{Server} is considered as a recent location. Otherwise, to find the last n' location(s), C_{Server} provides $MAC_{address}$ of detected outlier and its last detected location (L_{loc}) to A_{Server}. If L_{loc} is *null,* then query is passed to the XMPP server for further query to all *PSUs* (All_{PSU}) simultaneously and waits for their responses (R_{PSU}). When R_{PSU} arrive from the *PSUs*, they are aggregated and return to the A_{Server}.

Search the $MAC_{address}$ and find the L_{loc} from the C_{Server}
if $L_{loc} == \phi$ ***then***
  $R_{PSU} \leftarrow$ Query All_{PSU} in parallel
  Send R_{PSU} to A_{Server}
else
  Build a dynamic *k-d* tree for All_{PSU}
  $K_{PSU} \leftarrow$ find nearest $(n' - 1)$ *PSUs* of L_{loc} using *k*-NN
  $R'_{PSU} \leftarrow$ query to K_{PSU} and L_{loc}
  if $R_{\tilde{\phi}} == \phi$ ***then*** **// No record found**
    $R_{\phi} \leftarrow n'$
    Send R_{ϕ} to A_{Server}
  else if $R_{\tilde{\phi}} == 1$ ***then*** **// Only one record found**
    $R_{\phi} \leftarrow (n' - 1)$
    $C_{loc} \leftarrow$ loc. of *PSU* which provided data of $R_{\tilde{\phi}}$
    if $C_{loc} == L_{loc}$ ***then***
      Send R'_{PSU} and R_{ϕ} to A_{Server}
    else
      $K_{PSU} \leftarrow$ find R_{ϕ} nearest *PSUs* of C_{loc} by *k*-NN
      $R''_{PSU} \leftarrow$ query to K_{PSU} in parallel
      $R_{\phi} \leftarrow (n' - R''_{\phi} - 1)$
      Send R'_{PSU}, R''_{PSU} and R_{ϕ} to A_{Server}
  else **// More than one record found**

```
            R_ϕ ← (n′ - R_ϕ̃)
            Sort R′_PSU in descending order on the basis of time t
            RL_loc ← R′_PSU [0] // First index of an array
            if RL_loc == L_loc then
            |  Send R′_PSU and R_ϕ to A_Server
            else
            |  RL_loc ← R′_PSU [1] // Second index of an array
            |  K_PSU ← find R_ϕ nearest PSUs of RL_loc by k-NN
            |  R‴_PSU ← query to K_PSU
            |  R_ϕ ← (n′ - R_ϕ̃ - R‴_ϕ)
      — —   └─ Send R′_PSU, R‴_PSU and R_ϕ to A_Server
```

Fig. 4. LLTR algorithm

If a record already exists in the C_{Server} for the detected outlier, C_{Server} returns the L_{loc} of the $MAC_{address}$ and forwards the query to A_{Server} which further forwards it to XMPP server. Then, XMPP builds a dynamic *k-d* tree and searches the nearest (n' – 1) *PSUs* of L_{loc} using the *k*-NN approach. Now, XMPP server queries both list of *PSUs* recognized through *k*-NN and *PSU* at L_{loc} and waits for the response(s) (R'_{PSU}). If no information found, then XMPP server returns *null* to A_{Server}. If atleast one information found for the requested query, then XMPP server finds the current location (C_{loc}) of the *PSU* which provided that data. If C_{loc} is equal to L_{loc} then XMPP server sends the response to previous query (R'_{PSU}) with missing number of information (R_{ϕ}) calculated as (n' - 1) to A_{Server}. Now, A_{Server} will ask C_{Server} for the remaining R_{ϕ} locations. It will aggregate both XMPP server and C_{Server} response and return the aggregated response to the admin authorities.

If C_{loc} is not equal to L_{loc} then XMPP server finds the R_{ϕ} nearest *PSUs* of C_{loc} by *k*-NN and queries them in parallel. XMPP server collects the response (R''_{PSU}) and forwards R''_{PSU}, R'_{PSU} and R_{ϕ} (computed as (n - R''_{ϕ} - 1)) to the A_{Server}. If the information found in response is more than 1 then empty set is calculated as (n' - $R_{\tilde{\phi}}$). XMPP server sorts R'_{PSU} in descending order on the basis of *timestamp* and set the recent location (RL_{loc}) to location of the first *PSU* of the sorted response list. Now, if RL_{loc} is equal to L_{loc} , then XMPP server updates the RL_{loc} to the location of second *PSU* in the list of response. Moreover, XMPP server finds the R_{ϕ} nearest *PSUs* of RL_{loc} by *k*-NN, issues the query to these *PSUs* and waits for the response (R'''_{PSU}). Further, XMPP calculates the R_{ϕ} through (n'- $R_{\tilde{\phi}}$ - R'''_{ϕ}) and send the R'''_{PSU}, R'_{PSU}, and R_{ϕ} to A_{Server}.

The time complexity of building a *k-d* tree is O(n * log n) where n is the total number of *PSUs*. A search operation in *k-d* tree using *k*-NN takes O(k * log n) complexity. To search a record within a *PSU* takes only O(1) complexity as we are using *HashMap* for data storage. Therefore, if there are total m outliers detected, then the time complexity will be m * (O(n * log n) + O(k * log n))
=> m * (n + k) * O(log n).

6 Prototype Implementation Details

This section explains the detailed implementation of the *SmartISS* testbed.

We use a Jetson TK1 as a dynamic *PSU* [13] (see Fig. 5 (a)) which is NVIDIA's embedded Linux development kit having 192 CUDA cores, 2 GB RAM, 16 GB storage and Linux4Tegra OS. Jetson is basically used for the high-performance computing with low power consumption. It is most suitable for continuous operation under heavy workloads. We also use three different Smartphones (Samsung Galaxy S4 (SG_S4), Dell Venue 8 (DV_8) and Google Nexus 5 (GN_5)) as dynamic *PSUs* (see Fig. 5 (c)). Furthermore, a Linux-based desktop computer (Dell Precision T5600 system 64 GB RAM, Intel Xeon processor E5-2600 family, and 3 TB HDD) is used as a static sensing unit (deployed at UGPC lab, CSE Dept.) (see Fig. 5 (b)). For the location updates, we set the two properties *minimum time elapsed*, and *minimum distance* traveled to 1000 ms and 0, respectively. It means *PSU* will get the location update at every 1000 ms time interval. The location accuracy of a *PSU* is approximately equal either a *PSU* is static or moving at slow speed (i.e., walking) [47]. Moreover, it is very important to know that location accuracy keeps wandering even if a *PSU* is static at the same location and in an outdoor environment.

(a) Jetson TK1-based *PSU*

(b) Linux-based *PSU*

(c) Smartphone-based *PSU*

Fig. 5. Different types of *PSUs*

TABLE 2
HARDWARE SPECIFICATIONS OF SMARTPHONE-BASED *PSUs*

Device	Processing Capability	Cores (Family)	Wireless Features
SG_S4	Universal 5410 1.6GHz GPU: PowerVR SGX544MP3	4 (Cortax-A15)	Wi-Fi 802.11 a/b/g/n/ac, Dual-band, Wi-Fi Direct, GSM/HSPA, Bluetooth 4.0
GN_5	Qualcomm Snapdragon™ 800, 2.26 GHz, GPU: Adreno 330, 450MHz	4 (Krait-400)	Dual-band, Wi-Fi Direct, GSM/CDMA/HSPA/LTE, Wi-Fi 802.11 a/b/g/n/ac, Bluetooth 4.0
DV_8	Intel® Atom™ CPU Z3480 2.13 GHz	2 (Bay Trail)	Wi-Fi 802.11 b/g/n, GSM/HSPA, Bluetooth 4.0

Table 2 shows the hardware specifications of the

Smartphone-based *PSUs*. We use two types of external Wi-Fi adapters, Alfa AWUS036H [48], and a Tenda W311M [49]. These adapters provide the Wi-Fi scanning capability to the *PSUs*. The scanning range of Alpha and Tenda adapters are approximately 50–60 m, and 10–12 m, respectively. Alfa adapter-based *PSU* can detect smart devices within the range of 50-60 meters in Line-of-Sight (LoS) while, in crowded areas and other factors, such as microwaves, atmosphere, radio-frequency interference, buildings, metal construction, and trees, etc., a *PSU* can practically detect Smartphones within the range of 25-30 meters (radius) accurately. The Android version for *SG_S4*, *DV_8*, and *GN_5* are Lollipop (5.0.1), Kit Kat (4.4.4), and Marshmallow (6.0.1), respectively. Also, the battery capacities of *SG_S4*, *GN_5*, and *DV_8* are 2600 mAh, and 2300 mAh, and 4100 mAh, respectively. To handle the power consumption issue of dynamic *PSUs*, we use the Y-cable in the USB host mode to provide the support of an external battery.

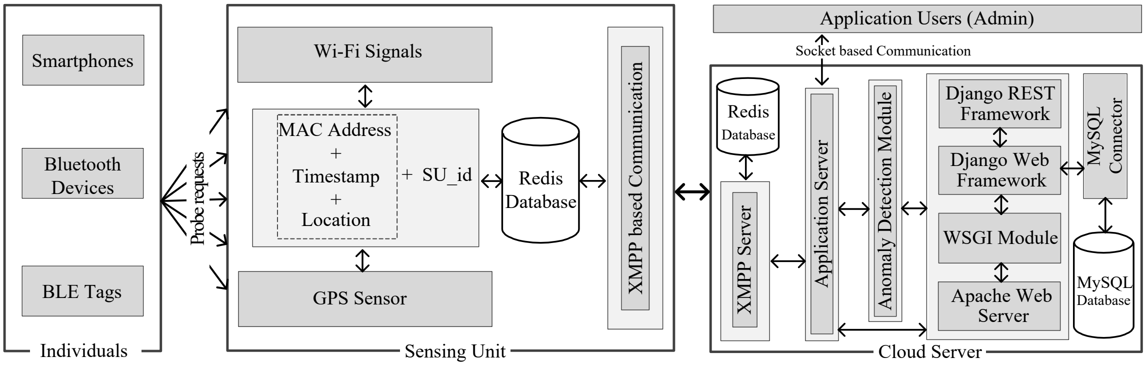
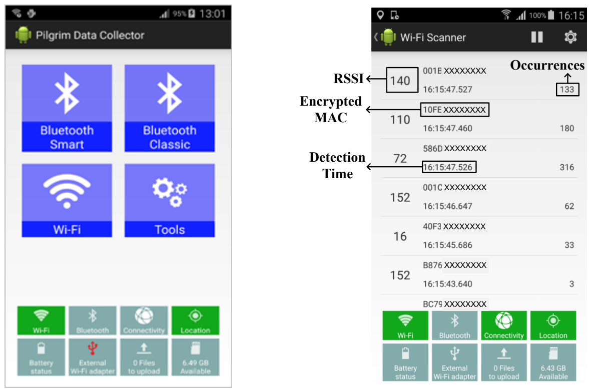


Fig. 6. Data flow among various modules of *SmartISS*

Fig. 6 shows the complete data flow of the *SmartISS*. To implement the C_{Server}, we use Intel® Core™ i7-3770 CPU @ 3.40 GHz, 64 GB RAM, and 2 TB storage. C_{Server} is implemented using Django web framework and MySQL. We develop anomaly detection module in Python.

All records from *PSUs* to C_{Server} are uploaded using the cURL HTTP POST method which supports multiple protocols, such as HTTP, FTP, HTTPS, SMTP, etc. The implementation of in-memory cache at both *PSU* and C_{Server} site is done using the Redis® [50] in-memory data structure store (database cache) to make the system time-efficient. Sometimes, it is also possible that a *PSU* stops uploading data to the C_{Server} due to some external factors, such as *PSU* software update, upload the erroneous data and upload data with low information. C_{Server} keeps checking that each *PSU* is uploading its data timely. C_{Server} also keeps monitoring the data quality and raises the alarm if uploaded data has lots of missing parameters and outliers.

Within the C_{Server}, we implement XMPP server and A_{Server}. XMPP server uses the XMPP framework which is based on client-server model. It uses XML which is an open source. A_{Server} is implemented using the Python. The Socket TCP protocol supports the communication between C_{Server} and A_{Server}. A_{Server} creates a new separate thread to handle each query concurrently.



(a) Main Activity (b) Wi-Fi Scanning

Fig. 7. User Interfaces of Smartphone-based *PSU*

(a) BLE and Bluetooth Tags (optional) (b) BLE/Bluetooth embedded Smartwatches (optional)

Fig. 8. Users' devices in case Smartphones are not available

A *PSU* can perform three types of scanning BLE, Bluetooth Classic (BC) or Wi-Fi (Fig. 7(a)). Fig. 7(b) represents the Wi-Fi scanning activity and the user interface which shows RSSI measurements, detected MACs, time of detections and detection frequency.

Users' devices are Smartphones having Wi-Fi and/or BT support. If users are without Smartphones, *SmartISS* can track individuals having BLE/BT tags, and BLE/BT enabled smartwatches (see Fig. 8 (a) and (b)).

7 Experimental Evaluation

The *SmartISS* system is developed from the scratch and extensively tested in both indoor and outdoor scenarios. In this section, we elaborate our dataset and experimental details with in-depth analysis of our results.

7.1 Background of prototype testbed

Experiments are performed in and around the Indian Institute of Technology, Roorkee (IITR) campus, India (Exp@IITR). IITR is a research and academic institute in Uttarakhand state, India. It has 1.48 km^2 area housing with several objects, such as administrative buildings, academic departments, library, student hostels, post office, schools, shops, banks, cafeterias canteen, etc.

Indoor localization means location logging and processing are performed inside the institute buildings, e.g., Library, Academic departments, etc., at IITR. Whereas, outdoor scenario means identifying and tracking individuals' locations and other information in open areas/grounds, roads/streets, and open space of buildings, etc. The indoor-outdoor scenario is the combination of the both above-mentioned scenarios in which buildings, roads, grounds, etc., are tracked in a random fashion.

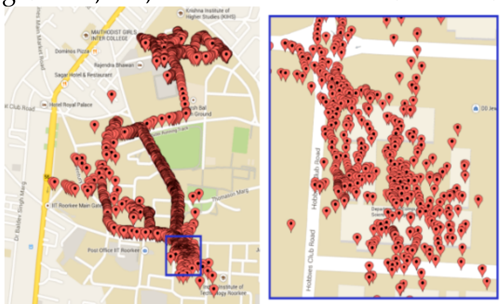


Fig. 9. Individuals' presence scanning using *SmartISS* for a particular path (MAC - Central Library - CSE Dept. - RB_Hostel - MAC) during Cognizance IITR (for 3 days)

Fig. 9 illustrates individuals' traces on the map using *SmartISS*. We deploy *SmartISS* system and then collect data for a technical festival of IITR, Cognizance - 2017 held on 24 - 26 March 2017. *SmartISS* track all those devices for which the Wi-Fi/BLE/BT is turned ON. We set all *PSUs* to monitor frames on channel 6. The maximum records uploaded per file are set to 100 and the duplicate detection rate of frames is fixed to 5 minutes which reduces the detection of same MAC ids upto a great extent. We have eight volunteers to collect the data using *PSUs* (4/4 volunteers for day/night). Volunteers keep moving most of the time in the specific regions and walk in and around campus.

Fig. 10 shows the event-wise individuals' traces on the map for three days during the Cognizance technical event. Fig. 11 shows the number of events held at the different locations of the IITR campus. As shown in Fig. 10, we deploy our *PSUs* in that pattern. One volunteer with the *PSU* keeps moving in the MAC area, one in the LBS ground area while one volunteer with *PSU* keeps moving around the Mgmt. Dept., LHC, and Hobby club. One volunteer keeps moving around the IITR campus in random fashion. The moving node for the campus can be a drone, robot, patrolling/security person, etc. One static *PSU* is deployed in the UGPC lab, CSE dept. building.

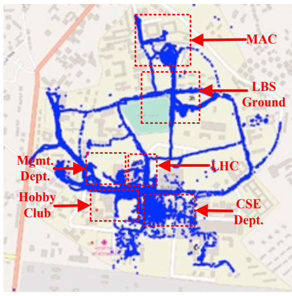


Fig. 10. *PSUs'* deployment in different areas and Individuals' traces collected during all 3 days' events of Cognizance at IITR

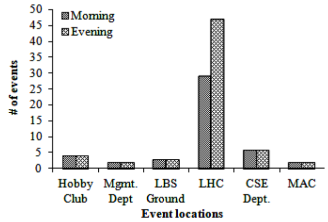


Fig. 11. Events statistics during Cognizance at IITR

In our database, total records stored are approximately 117,121 in which the unique MAC records are 14,672. We collect data through Wi-Fi and BT scanning where total traces collected through Wi-Fi and BT are 116,089 and 1,032, respectively. Total records consist of data from both static and dynamic *PSUs*.

7.2 Dataset analysis

We perform some basic analysis on the captured data, like the total number of individuals detected by a *PSU*, day-wise and time-wise presence of individuals at the different event locations, and distribution of individuals in the various events.

We further analyze the collected data to find the mobility pattern of the people during the event. Fig. 12 (a) shows the total MAC ids detected by all sensing units in and around the IITR campus for the entire time span of 3 days. Fig. 12 (b) and (c) show the day-wise and hour-wise distribution of distinct MAC ids, respectively during the event. Highest visiting frequencies detected for the morning and evening sessions are day-1 with 6,880 and day-2 with 2,900 number of individuals, respectively. The lowest visiting frequency in and around the IITR campus is during the time 18-20 (24-hours format) for all three days. Fig. 12 (d) shows the total number of individuals present at the different event locations. Hobby Club, and Mgmt. Dept. are the locations where detection of individuals' MAC addresses is high in compared to other locations. It also shows the average pause time which is average time spent by the individuals at that particular location. We extend our analysis to find the total number of individuals and a common number of individuals present in 3 days (Fig. 12 (e) and (f)).

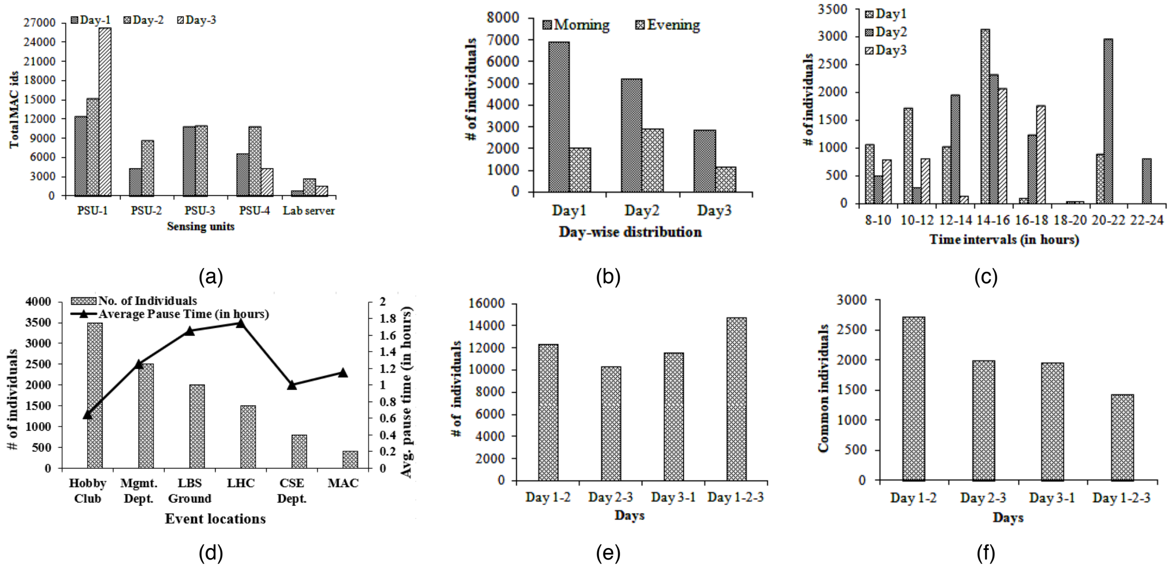


Fig. 12. Data Analysis during Cognizance Technical Festival through 4-dynamic and 1-static *PSU*(s)

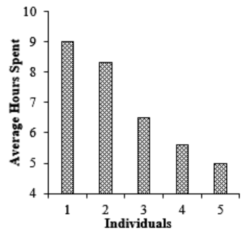


Fig. 13. Top 5 individuals for spending maximum time in and around events locations during Cognizance at IITR

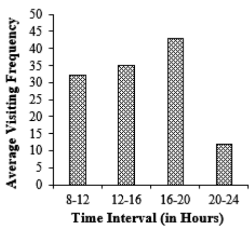


Fig. 14. Average visiting frequency of an individual in IITR campus during the time interval (in hours), Cognizance, IITR

Fig. 13 shows the average hours spent by the top 5 individuals in and around the event locations. The visiting frequency sequences of an individual in and around the event locations for the fixed time intervals is shown in Fig. 14. It also shows the working pattern and hours spent in the events. The selected person spends more time during the day hours of 12-16 and 16-20.

To further find the visiting patterns of an individual, we draw the day/hour-wise heatmap of an individual's visiting frequency in the events (see Fig. 15). The analysis depicts that the individual remains highly active during the period of 19-21 (2 hours) on the first day of cognizance festival. Fig. 16 shows the visualization of the trajectory of an individual during Cognizance event at IITR.

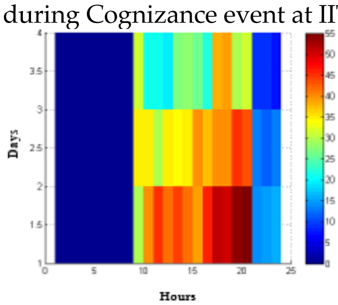
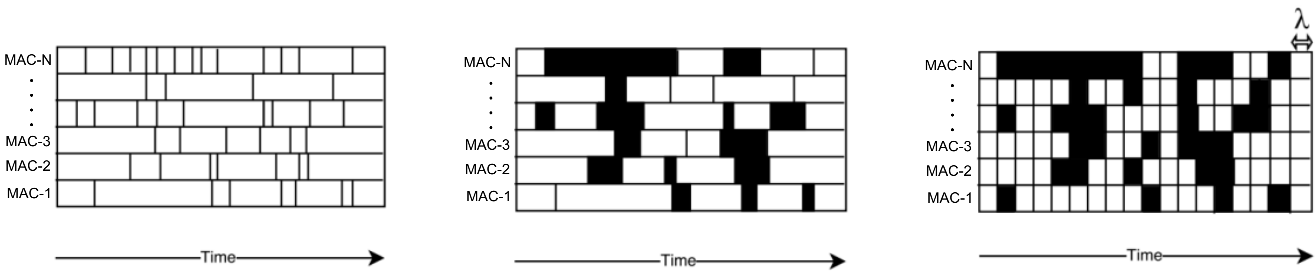


Fig. 15. Day-Hour wise visiting frequency of an individual in the IITR campus during Cognizance event for 3 days

Fig. 16. Trajectory (RB_Hostel - LHC - Mgmt._Dept. - CSE Dept.) of an individual during Cognizance on day-1 retrieved from database



(a) Time Point Sequence
(b) Time Interval Sequence
(c) Discretized Time Interval Sequence

Fig. 17. Data preprocessing

We inspect the time delay between all consecutive probe requests from same device (Samsung, Apple iPhone, Google Nexus, Dell, etc.) in the direct line of sight which are stationary and within the range of the adapter and realize that for more than 95% of packets time delay between the current packet and the previous packet is below 15 minutes. So, to achieve a false negative rate of less than 5%, we set a segment time *st* equal to 15 minutes. If devices are not placed in the direct line of sight and are not static, then the average rate of false negatives rate for *st* equal to 15 minutes increases to 38% due to packet loss.

Fig. 17 (a) shows Time Point Sequences (PS), where each vertical bar (|) shows the instantaneous timestamp when the probe requests corresponding to the MAC address *m* is received. We can observe that many probe requests are received in short time span. However, storing information about closely spaced probe requests from each *m* is redundant and costly. As we are more interested in analyzing the time interval during which the smart device was present in the vicinity, we obtain Time Interval Sequence (IS) for every MAC address *m* from it's corresponding *PS*.

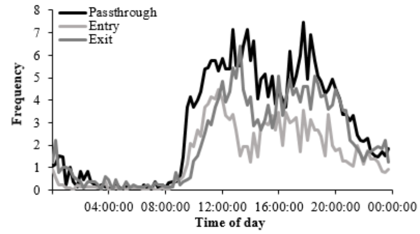
Fig. 17 (b) shows *IS*, where each filled rectangle shows the time interval when the probe request corresponding to the MAC address *m* is received. To obtain *IS* from *PS*, we inspect the time delay between consecutive probe requests in *PS*. If the time delay is below a threshold *st*, then the later packet extends the previous interval otherwise it spawns a new interval. The *ISs* reduce the storage requirement. However, performing piece-wise analysis on *IS* is difficult as the intervals are of varying length (i.e., duration). So, the *ISs* are discretized into sequence of regularly sampled equal sized unit intervals of length λ. We set λ equal to $st/2$ so that any gap between intervals which is more than *st* minutes is captured. Fig. 17 (c) shows Discretized Interval Sequences (DIS).

The statistics, such as average stay time, number of visitors, and frequency reveal semantics of a particular location [51]. Thus, we deploy a *PSU* in the UGPC lab, CSE Dept. to find out the average day-wise visiting pattern of frequent visitors. MAC addresses are classified into three groups: *new visitor*, if a visitor is traced for the first time on a particular day, *frequent visitor* who frequently detected by nearby *PSUs* in the time span of a day, and *non-frequent visitor* who is not detected in a day. Generally, these visitors are outsiders who visit only occasionally, such as during special events.

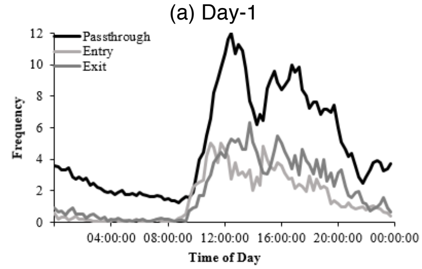
To understand the location semantics, we apply feature engineering on the collected Wi-Fi records. Fig. 18 (a), (b), and (c) show the day-wise pattern for frequent visitors at CSE Dept. For each time segment of λ, we further categorize each sub-group into *Passthrough* (if visitors pass from the location before, during and after the slot), *Entry* (if visitor has arrived at the venue during the given slot), and *Exit* (if visitor left the venue).

For analyzing this, we need to track an individual's MAC id for *N* time slots before and after the current time slot.

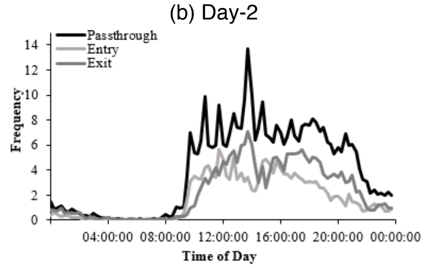
1. A MAC id is a *Passthrough* if it is not detected in the *N* time slots before or after the given slot but detected only during the given time slot.
2. A MAC id is considered as *Entry* at a location if it is not detected in the *N* time slots before the given time slot but detected during the given time slot and in any of the *N* time slots after the given time slot.
3. A MAC id is called *Exit* at a location if it is detected in some of the *N* time slots before the given time slot and also detected during the given time slot but not in any of the *N* time slots after the given slot.



(a) Day-1



(b) Day-2



(c) Day-3

Fig. 18. Average visiting frequency by top 5 frequent visitors for 3 days at UGPC lab, CSE Dept. during Cognizance

7.3 Information retrieval from APs

We further broaden our analysis and use an Android-based *APlogger* application [52] for finding the mobility pattern of individuals in the IITR campus. *APlogger* installed on Smartphones maintains the traces of *APs* available in the surrounding region with the current GPS location of Smartphone and the timestamp at which *APs* are detected.

We install *APlogger* on four Smartphones (*GN_5*, *SG_S4*, and *two DV_8*). A record in *APlogger* maintains the following fields: *<Lat., Long., timestamp, Smartphone model, SSID, BSSID, RSSI, Security mode, frequency>* where *Lat.* and *Long.* are the GPS location of a Smartphone having *APlogger*, *timestamp* is the time at which an *AP* location is traced, *model* is the model of the Smartphone used for *AP* sensing, *SSID* is the *AP's* name, *BSSID* is the MAC address of an *AP*, *RSSI* is the signal strength of detected *AP*, *security mode* is the type of security used in *AP* and *frequency* shows the signal frequency of an *AP*. However, we use only three fields for the analysis, *<Lat., Long.>*, *timestamp*, and *SSID*.

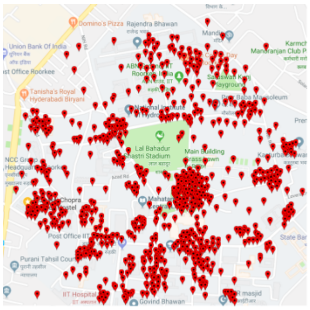


Fig. 19. Plotting of Wi-Fi hotspots discovered on Google Map

Fig. 20. Location of top 20 individuals in IITR during Cognizance using *APlogger*

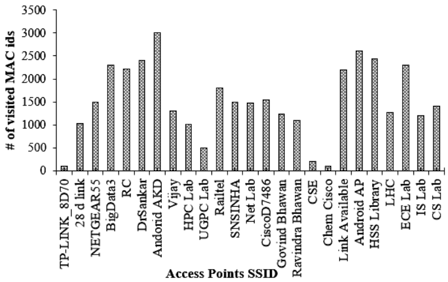


Fig. 21. Number of individuals detected by different *APs* during Cognizance

Student volunteers with the Smartphones having *APlogger* app walked in and around the IITR campus for 15 days and collect a list of 733 *APs* (hotspots). We plot the total number of detected *APs* on the Google Map as shown in Fig. 19. To estimate the *AP's location*, we average the location of all the occurrences of an *SSID* in the collected data. As shown in Fig. 19, the location with a higher concentration of Wi-Fi hotspots are offices, departments, and laboratories areas as these infrastructures require Wi-Fi networks for Internet usage. Residential areas have less number of Wi-Fi networks. On the other hand, playgrounds or open areas have almost no Wi-Fi networks.

To find the visiting pattern in the IITR campus during Cognizance festival, we find top 20 most frequently visited individuals from the collected data and plot their traces on Google map (see Fig. 20). We also analyze the visited number of smart devices (MACs) at each *AP* using the *APlogger* app and *PSUs* as shown in Fig. 21. Through the analysis of probe requests, we collect 1,41,695 probe records (72,802 are directed probes, and 68,893 are broadcasted) in total through the *PCAP* app on *PSUs*.

7.4 Simulation results

The generation of probe requests depends on several factors, such as device OS, version, the model, etc., [36]. Fig. 22 shows the number of probe request generation whenever a Smartphone is *associated* (A) and *not associated* (NA) with any *AP*. The result shows that a Smartphone generates less number of probe requests when Smartphone's display screen is *locked*, and it is *A* to an *AP*. On the other hand, generation of probe requests are high when Smartphone's display is *on*, and the phone is *NA* with any *AP*. Results also represent that Smartphones, such as *GN_5* and *SG_S4* generate probe requests even they are not *A* to any *AP*, and their display screens are *off*.

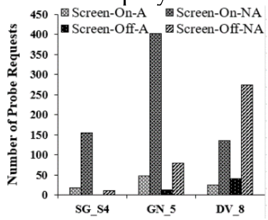


Fig. 22. Number of probe requests generated when a mobile is A / NA with *AP*

We conduct experiments to observe the performance of Jetson TK1 based *PSU* against frame processing success ratio (Rsucc) which is a ratio between successfully processed frames (generated and sent by all smart devices in the vicinity of sensing area) at a *PSU* to the total number of frames generated by all smart devices covered by the same *PSU*.

For simulation experiment, we develop a Linux-based Wi-Fi frame injector using the PCAP library [53]. The Wi-Fi frame injector is installed on Dell Precision T5600 system having 64 GB RAM, Intel Xeon processor E5-2600 family, and 3 TB HDD. The injector works as individual devices to test the processing capability of the *PSU*. The injector adjusts the following parameters: MAC ids (same and / or different), the delay between the frames, number of frames to transmit, types of frames (Data & Request to Send (RTS)). We use Jetson TK1 as a *PSU* to capture and intercept the frames transmitted from nearby individual's devices.

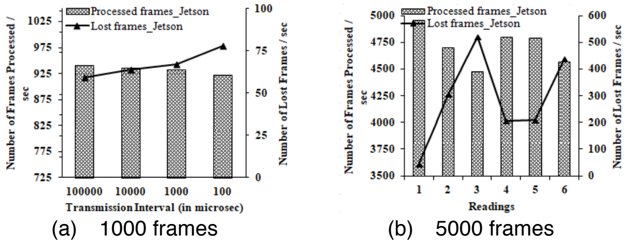


Fig. 23. *PSU's* processing capability with same MAC ids

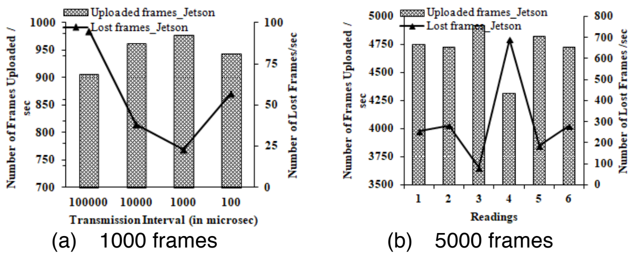
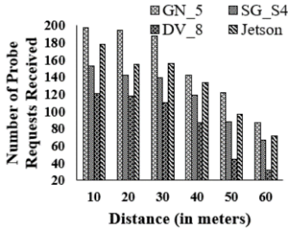


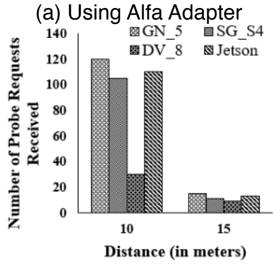
Fig. 24. *PSU's* uploading capability with unique MAC ids

To analyze the processing capability of a *PSU*, Wi-Fi injector transmits 1000 frames/sec with the same MAC ids at the different transmission intervals (100 to 100,00 microseconds) (see Fig. 23 (a)). After that, Wi-Fi injector generates 5000 frames/sec having same MAC ids at the fixed transmission interval (see Fig. 23 (b)). The average number of frames captured by a *PSU* are 933 and 4712 for 1000 and 5000 frames/sec, respectively. Therefore, the *Rsucc* of Jetson is 93.3% and 94.24% for 1000 and 5000 frames/sec, respectively. Another experiment is performed for the

1000 and 5000 frames/s having unique MAC ids at different transmission intervals ranging from 100 to 100,000 micro-seconds (see Fig. 24 (a)) and at the fixed transmission (see Fig. 24 (b)), respectively, to find the *PSU* capability of handling the numbers of individuals' devices at the same time. The average number of frames uploaded by the *PSU* are 932 out of 1000 and 4704 out of 5000. Therefore, the *Rsucc* of Jetson is 93.2% for 1000 frames and 94.08% for 5000 frames.



(a) Using Alfa Adapter



(b) Using Tenda Adapter

Fig. 25. Impact of the distance on the probe requests received by *PSUs*

To find the impact of distance on the number of probe requests received by *PSUs*, we perform an experiment for 10 mins for Alfa and Tenda adapters as shown in Fig. 25 (a) and (b), respectively. The more distance decreases the probability of an individual's device detection but still that number is enough sufficient to detect the presence of individuals by *PSUs*.

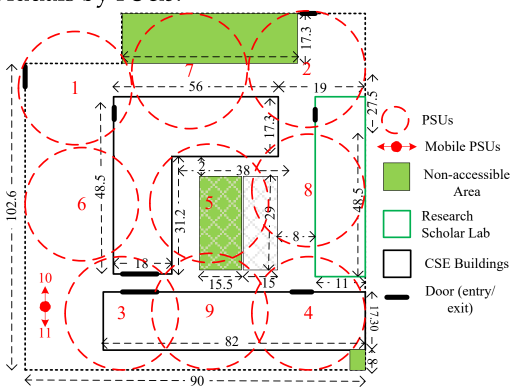


Fig. 26. *PSUs* deployment to find the impact of number of *PSUs* and their scanning range on the system performance

Furthermore, to find the impact of the number of *PSUs* and their scanning range on the system performance, such as number of outlier detection, we establish a scenario in the CSE Dept., IITR. We divide the whole area of CSE building in 30 x 30 m grid cells. We take two sensing range of a *PSU*: 30 meters (Alfa Adapter) and 10 meters (Tenda Adapter). We take 22 *PSUs* and deploy in the CSE Dept. building indoor (only ground floor) and outdoor as shown in Fig. 26. *PSUs* 1-9 are static while *PSUs* 10-11 keep moving in the clockwise and anti-clockwise direction, respectively.

We track individuals in the CSE building for a week (04 - 10 Sep., 2018) during the peak time period 11:00 - 12:00. For finding the impact of the number of *PSUs*, we take three cases: Case 1 (C1_R) having the outliers detected by 1-5 *PSUs*, Case 2 (C2_R) having the outliers detected by 1-5 and 10-11 *PSUs* and Case 3 (C3_R) having the outliers detected by all *PSUs* where *R* represents the *PSU's* scanning range 30/10 m. We perform all experiments at the same time.

The total number of records collected through this experiment are 1007, 1311, and 1685 for the cases *C1_30*, *C2_30* and *C3_30*, respectively. For the cases *C1_10*, *C2_10*, and *C3_10*, total number of records collected are 550, 645, and 862, respectively. The unique number of records uploaded at the C_{Server} are 209, 253, and 253 for the cases *C1_30*, *C2_30*, and *C3_30*, respectively and 104, 189, and 206 for the cases *C1_10*, *C2_10*, and *C3_10*, respectively.

The outlier detection for the scanning range 30 m are 11, 15 and 15 for the Case *C1_30*, *C2_30*, and *C3_30*, respectively while for scanning range 10 m, outlier detected are 6, 9 and 11 for the Case *C1_10*, *C2_10*, and *C3_10*, respectively. The number of outlier detection rate for the case *C2_30* and *C3_30* are equal while the case *C2_10* and *C3_10* have a difference of 2 outliers. For the high scanning range, *SmartISS* is able to capture equal number of outliers for *PSUs'* deployment scenarios in *C2_30* and *C3_30* while in case of low scanning range, the performance of the *SmartISS* is reducing w.r.to outliers' detection for similar *PSUs'* deployment scenarios. Through these results, we can observe that even though two *PSUs* are deployed at the end of a road segment (having no exit/turn in middle), they must be covering the entire moving path (width) of their own deployed area.

Increased number of *PSUs* are reducing the efforts to capture the data and increasing the tracking data of users. As far as the whole area is covered through sparse deployment of static and mobile *PSUs*, more number of *PSUs* than that will increase the duplicate detection of the individuals' traces for *SmartISS*. Even though it is known that the more tracked data you have, it is better, but data redundancy can reduce the data quality if not handled properly.

7.5 Performance Metrics

In this section, we formally discuss the performance metrics used for evaluating the proposed *SmartISS* system.

1. **PSU Selection Accuracy (P_A):** It is the selection of the optimal number of *PSU(s)* from the total number of *PSUs* for retrieving the requested data.
   P_A= (Optimal number of *PSUs* selected / Total number of *PSUs*) * 100
2. **Response Time:** It is the time interval between the instant at which the query is sent and the moment at which the A_{Server} receives corresponding response.

7.6 Experiment@IITR

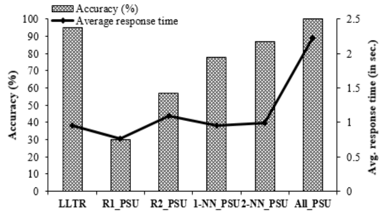
We perform the experiments into two categories: *Exp@OutlierDetection* and *Exp@TrajectoryAnalysis* for evaluating the features and functionalities of *SmartISS* in real time.

Exp@OutlierDetection

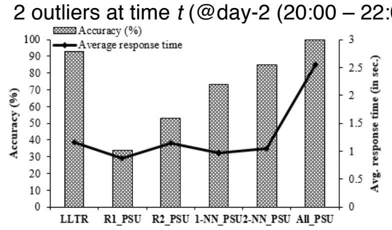
In this experiment, the average time taken to process and find out an anomaly from the location data through *k-d* tree with *k*-NN is 0.110 msec. For better understanding and to show the efficacy of the *SmartISS* in real time, we find outliers for a music event in LBS ground where approximately 2000 individuals gathered at the day-2 (20:00 – 22:00 time). *SmartISS* detects 2 outliers at time 20:19:33. To find the scalability of the *SmartISS*, we calculate the response time and accuracy for the outliers detected during the 3 days of the technical festival.

Exp@TrajectoryAnalysis

In order to analyze the performance of the proposed LLTR algorithm w.r.to other similar schemes, we pick the five PSU-selection approaches for finding the current location of the detected outlier(s): *Random 1 PSU* (R1_PSU), *Random 2 PSU* (R2_PSU), *1-NN PSU* (1-NN_PSU), *2-NN PSU* (2-NN_PSU) and all *PSU* selection (All_PSU). In *R1_PSU*, we select one *PSU* randomly among the list of *PSUs* to find the current location of the requested outlier while, in *R2_PSU*, we select two *PSUs* randomly. In *1-NN_PSU*, XMPP server finds the 1-nearest neighbor of the requested MAC address and queries to both 1-nearest *PSU* and the *PSU* which uploaded location of requested MAC most recently to the C_{Server}. While, in *2-NN_PSU*, 2-nearest neighbors are selected instead of one. In *All_PSU*, all *PSUs* are selected.



(a) 2 outliers at time *t* (@day-2 (20:00 – 22:00 time))



(b) 98 outliers in 3 days

Fig. 27. *PSU*(s) selection accuracy and average response time of different approaches for finding latest location (for *n'* = 3) of 98 outliers

Fig. 27 (a) and (b) show the comparison of the *PSU* selection accuracy of various approaches to find the latest location of the outlier. To find the efficacy of the system, we analyze *SmartISS* performance only in an event in which 2 outliers are detected at a time *t* (where *t* = 20:19:33 as discussed above) (see Fig. 27 (a)). The *PSU* selection accuracy of *LLTR*, *R1_PSU*, *R2_PSU*, *1-NN_PSU*, *2-NN_PSU* and *All_PSU* are 95.3%, 30.3%, 57.4%, 78.1%, 87.2% and 100%, respectively. The average response time of *LLTR*, *R1_PSU*, *R2_PSU*, *1-NN_PSU*, *2-NN_PSU*, and *All_PSU* are 0.962 sec., 0.763 sec., 1.13 sec., 0.961 sec., 0.994 sec., and 2.231 sec., respectively.

Furthermore, we extend our analysis to check the scalability and accuracy of *SmartISS*, we pass the 98 queries at the same time (98 outliers for three days) to the A_{Server} (see Fig. 27 (b)). The *PSU* selection accuracy of *LLTR*, *R1_PSU*, *R2_PSU*, *1-NN_PSU*, *2-NN_PSU* and *All_PSU* are 93.2%, 34.1%, 53.5%, 72.1%, 85.1% and 100%, respectively. The average response time of *LLTR*, *R1_PSU*, *R2_PSU*, *1-NN_PSU*, *2-NN_PSU*, and *All_PSU* are 1.153 sec., 0.878 sec., 1.137 sec., 0.974 sec., 1.045 sec., and 2.556 sec., respectively. The time taken by *1-NN_PSU* is equal to *R1_PSU* as both are querying only one *PSU*. The time taken by *2-NN_PSU* is more than *1-NN_PSU* and *R1_PSU* while less than *All_PSU*. Moreover, the average response time of *LLTR* is reduced to 25.89% compared to *All_PSU*.

Although *All_PSU* approach has 100% accuracy for finding the latest location of 98 outliers while the average response time is high in compared to other approaches. The proposed algorithm, *LLTR* has low response time for finding latest location at the expense of only a little difference in accuracy. Experimental results show that this *trade* is worthy.

8 Conclusion and Future Works

In this paper, we designed and implemented a novel intelligent surveillance system (named, SmartISS) for public safety. *SmartISS* collects the unique MAC ids of the individuals emitted from their wireless devices and uses an outlier detection algorithm to detect individual(s) mobility against the normal behavior of the crowd. The outlier information is further used to find the recent locations of the suspicious person. It is not sufficient to query only the C_{Server} for finding the latest locations of suspicious person as C_{Server} can have outdated data. Therefore, we proposed an algorithm to select the optimal number of sensing units deployed at geographically dispersed locations. To validate and to show the usability of *SmartISS*, we developed a real prototype testbed and evaluated it extensively in both indoor and outdoor environments on a real-world dataset of more than 117,121 traces collected during the technical festival, Cognizance 2017 held at IIT Roorkee campus, India.

On a broader canvas, the *SmartISS* demonstrated the efficacy of sensing units for the surveillance of individuals in both indoor and outdoor scenarios. *SmartISS* provided a high level of accuracy and insights which participatory mobile sensing can not achieve in gatherings, such as congregations, rallies. We believe that many other insights of practical interest (e.g., frequent region detection, quantifying how many individuals can visit the specific location in the near future, etc.) can be estimated using the *SmartISS* system. Moreover, our results show that certain aggregate insights (e.g., events popularity, flow of the mass) can be accomplished even with very low levels of analysis.

In future, we shall extend this work for large-scale scenarios, such as vehicular traffic control, evacuation path planning, and post-disaster recovery.

REFERENCES

[1] J. Shen, J. Cao, X. Liu, and C. Zhang, "Dmad: Data-driven measuring of Wi-Fi access point deployment in urban spaces," *ACM Trans. Intell. Syst. Technol.*, vol. 9, no. 1, pp. 1–29, Aug. 2017.

[2] R. L. Hughes, "THE FLOW OF HUMAN CROWDS," *Annu. Rev. Fluid Mech.*, vol. 35, no. 1, pp. 169–182, Jan. 2003.

[3] N. Smelser, *Theory of collective behavior*. Quid Pro Books, 2011.

[4] C. Yang and H. R. Shao, "WiFi-based indoor positioning," *IEEE Commun. Mag.*, vol. 53, no. 3, pp. 150–157, 2015.

[5] G. De Angelis, G. Baruffa, and S. Cacopardi, "GNSS/cellular hybrid positioning system for mobile users in Urban scenarios," *IEEE Trans. Intell. Transp. Syst.*, vol. 14, no. 1, pp. 313–321, 2013.

[6] L. M. Ni, D. Zhang, and M. R. Souryal, "RFID-based localization and tracking technologies," *IEEE Wirel. Commun.*, vol. 18, no. 2, pp. 45–51, 2011.

[7] P. C. Deepesh, R. Rashmita, A. Tiwari, and V. N. Rao, "Experiences with using iBeacons for Indoor Positioning," in *9th India Software Engineering Conference. ACM*, 2016, pp. 184–189.

[8] S. Gong, C. C. Loy, and T. Xiang, "Security and surveillance system," in *Visual Analysis of Humans*, Springer, 2011, pp. 455–472.

[9] A. Ben Mabrouk and E. Zagrouba, "Abnormal behavior recognition for intelligent video surveillance systems: A review," *Expert Syst. Appl.*, vol. 91, pp. 480–491, 2017.

[10] L. Ren, X. Chen, B. Xie, Z. Tang, T. Xing, C. Liu, W. Nie, and D. Fang, "DE 2: localization based on the rotating RSS using a single beacon," *Wirel. Networks*, vol. 22, no. 2, pp. 703–721, 2016.

[11] T. Im and P. De, "User-assisted OCR on outdoor images for approximate positioning," in *Lecture Notes in Electrical Engineering*, 2016, vol. 376, pp. 1419–1429.

[12] A. Boukerche, H. A. B. Oliveira, E. F. Nakamura, and A. A. F. Loureiro, "Secure localization algorithms for wireless sensor networks," *IEEE Commun. Mag.*, vol. 46, no. 4, pp. 96–101, 2008.

[13] "Jetson TK1," 2017. [Online]. Available: http://www.nvidia.com/object/jetson-tk1-embedded-dev-kit.html. [Accessed: 30-Nov-2017].

[14] "Location and context overview: Google Location API." [Online]. Available: https://developer.android.com/training/location/. [Accessed: 17-Sep-2018].

[15] M. Wirz, T. Franke, D. Roggen, E. Mitleton-Kelly, P. Lukowicz, and G. Tröster, "Probing crowd density through smartphones in city-scale mass gatherings," *EPJ Data Sci.*, vol. 2, no. 1, pp. 1–24, Dec. 2013.

[16] N. Koshak and A. Fouda, "Analyzing pedestrian movement in mataf using gps and gis to support space redesign," in *9th international Conference on Design and Decision Support systems in Architecture and Urban Planning*, 2008, no. July, pp. 1–14.

[17] A. B. M. Musa and J. Eriksson, "Tracking unmodified smartphones using wi-fi monitors," in *Proceedings of the 10th ACM Conference on Embedded Network Sensor Systems - SenSys '12*, 2012, p. 281.

[18] I. Rose and M. Welsh, "Mapping the urban wireless landscape with Argos," *Proc. 8th ACM Conf. Embed. Networked Sens. Syst. - SenSys '10*, p. 323, 2010.

[19] N. Cheng, P. Mohapatra, M. Cunche, M. A. Kaafar, R. Boreli, and S. Krishnamurthy, "Inferring user relationship from hidden information in WLANs," *Proc. - IEEE Mil. Commun. Conf. MILCOM*, 2012.

[20] H. Hong, C. Luo, and M. C. Chan, "SocialProbe: Understanding Social Interaction Through Passive WiFi Monitoring," in *Proceedings of the 13th International Conference on Mobile and Ubiquitous Systems: Computing, Networking and Services - MOBIQUITOUS 2016*, 2016, pp. 94–103.

[21] M. V. Barbera, A. Epasto, A. Mei, V. C. Perta, and J. Stefa, "Signals from the crowd: Uncovering social relationships through smartphone probes," *Proc. 2013 Conf. Internet Meas. Conf.*, pp. 265–276, 2013.

[22] B. Bonné, A. Barzan, P. Quax, and W. Lamotte, "WiFiPi: Involuntary tracking of visitors at mass events," in *World of Wireless, Mobile and Multimedia Networks (WoWMoM) 14th International Symposium and Workshops*, 2013, pp. 1–6.

[23] A. Di Luzio, A. Mei, and J. Stefa, "Mind your probes: De-anonymization of large crowds through smartphone WiFi probe requests," in *Proceedings of IEEE INFOCOM*, 2016.

[24] Y. Wang, J. Yang, Y. Chen, H. Liu, M. Gruteser, and R. P. Martin, "Tracking human queues using single-point signal monitoring," in *Proceedings of the 12th annual international conference on Mobile systems, applications, and services - MobiSys '14*, 2014, pp. 42–54.

[25] P. Bahl and V. N. Padmanabhan, "RADAR: an in-building RF-based user location and tracking system," in *Proceedings IEEE INFOCOM 2000. Conference on Computer Communications. Nineteenth Annual Joint Conference of the IEEE Computer and Communications Societies (Cat. No.00CH37064)*, vol. 2, pp. 775–784.

[26] M. Versichele, T. Neutens, M. Delafontaine, and N. Van de Weghe, "The use of Bluetooth for analyzing spatiotemporal dynamics of human movement at mass events: A case study of the Ghent Festivities.," *Appl. Geogr.*, vol. 32, no. 2, pp. 208–220, 2012.

[27] J. M. Oh, N. Moon, and S. Hong, "Trajectory based database management for intelligent surveillance system with heterogeneous sensors," *Multimed. Tools Appl.*, vol. 75, no. 23, pp. 15429–15444, Dec. 2016.

[28] J. C. B.Hofmann-Wellenhof, H.Lichtenegger, *Global Positioning System_ Theory and Practice - B*, vol. 28, no. 1–2. Springer Vienna, 2012.

[29] L. Yang, J. Cao, W. Zhu, and S. Tang, "Accurate and efficient object tracking based on passive RFID," *IEEE Trans. Mob. Comput.*, vol. 14, no. 11, pp. 2188–2200, Nov. 2015.

[30] F. Subhan, H. Hasbullah, A. Rozyyev, and S. T. Bakhsh, "Indoor positioning in Bluetooth networks using fingerprinting and lateration approach," *2011 Int. Conf. Inf. Sci. Appl. ICISA 2011*, 2011.

[31] H. S. Maghdid, I. A. Lami, K. Z. Ghafoor, and J. Lloret, "Seamless Outdoors-Indoors Localization Solutions on Smartphones," *ACM Comput. Surv.*, vol. 48, no. 4, pp. 1–34, 2016.

[32] F. Adamsky, T. Retunskaia, S. Schiffner, C. Köbel, and T. Engel, "WLAN Device Fingerprinting using Channel State Information (CSI)," in *Proceedings of the 11th ACM Conference on Security & Privacy in Wireless and Mobile Networks - WiSec '18*, 2018, pp. 277–278.

[33] Xuyu Wang, Lingjun Gao, Shiwen Mao, and S. Pandey, "DeepFi: Deep learning for indoor fingerprinting using channel state information," *2015 IEEE Wirel. Commun. Netw. Conf.*, no. October, pp. 1666–1671, 2015.

[34] X. Tian, S. Zhu, S. Xiong, B. Jiang, Y. Yang, and X. Wang, "Performance Analysis of Wi-Fi Indoor Localization with Channel State Information," *IEEE Trans. Mob. Comput.*, vol. XX, no. c, p. 1, 2018.

[35] LAN/MAN Standard Committee, "Wireless LAN medium access control (MAC) and physical layer (PHY) specifications, 1999 edition," 1999.

[36] T. Kulshrestha, D. Saxena, R. Niyogi, V. Raychoudhury, and M. Misra, "SmartITS: Smartphone-based identification and tracking using seamless indoor-outdoor localization," *J. Netw. Comput. Appl.*, vol. 98, pp. 97–113, 2017.

[37] "KD Tree for fast generalized N-point problems," 2017. [Online]. Available: http://scikit-learn.org/stable/modules/generated/sklearn.neighbors.KDTree.html. [Accessed: 25-Nov-2017].

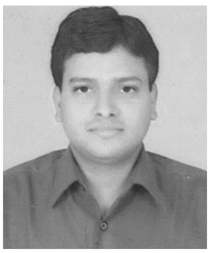
[38] D. T. Larose, "k-nearest neighbor algorithm," in *Discovering Knowledge in Data: An Introduction to Data Mining*, 2005, pp. 90–106.

[39] "Single detection in 2D dimension | Software Programming." [Online]. Available: https://kunuk.wordpress.com/2013/01/13/single-detection-in-2d-dimension/. [Accessed: 23-Sep-2018].

[40] A. Hornsby and R. Walsh, "From instant messaging to cloud computing, an XMPP review," in *Proceedings of the International Symposium on Consumer Electronics, ISCE*, 2010, pp. 1–6.

[41] D. Yang, G. Xue, X. Fang, and J. Tang, "Crowdsourcing to smartphones: incentive mechanism design for mobile phone sensing," in *Proceedings of the 18th annual international conference on Mobile computing and networking - Mobicom '12*, 2012, p. 173.

[42] D. Drohan, "Google IO Presentation-Fused Location Provider and Priority Modes." [Online]. Available: https://ddrohan.github.io/msc-mad/topic04-google-services/talk-2-google-2/01.location.pdf. [Accessed: 17-Sep-2018].
[43] R. Rise, S.-H. Cho, and D. Kaylor, "RC4 Encryption," pp. 4–7, 1994.
[44] D. Gentry and A. Pennarun, "Passive Taxonomy of Wifi Clients using MLME Frame Contents," Aug. 2016.
[45] 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," vol. 2017, no. 4, pp. 365–383, 2017.
[46] M. Vanhoef, C. Matte, M. Cunche, L. S. Cardoso, and Piessens Frank, "Why MAC Address Randomization is not Enough: An Analysis of Wi-Fi Network Discovery Mechanisms," *Asia Ccs*, pp. 413–424, 2016.
[47] A. SREDER, "Android: GPS positioning and location strategies - codecentric Blog," 2014. [Online]. Available: https://blog.codecentric.de/en/2014/05/android-gps-positioning-location-strategies/. [Accessed: 23-Nov-2017].
[48] "Alfa Wireless Adapter (AWUS036H)," 2017. [Online]. Available: http://www.alfa.com.tw/products_show.php?Pc=34&ps=92. [Accessed: 25-Nov-2017].
[49] "Tenda Wireless Adapter," 2017. [Online]. Available: http://www.tenda.cn/en/product/category-23.html. [Accessed: 25-Nov-2017].
[50] "Redis in-memory data structure store," 2017. [Online]. Available: http://redis.io/. [Accessed: 15-Dec-2017].
[51] T. Kulshrestha, D. Saxena, R. Niyogi, M. Misra, and D. Patel, "An Improved Smartphone-Based Non-Participatory Crowd Monitoring System in Smart Environments," in *International Conference on Computational Science and Engineering and IEEE/IFIP International Conference on Embedded and Ubiquitous Computing, CSE and EUC 2017*, 2017, vol. 2, pp. 132–139.
[52] T. Kulshrestha, R. Niyogi, and D. Patel, "A fast and scalable crowd sensing based trajectory tracking system," in *Tenth International Conference on Contemporary Computing (IC3)*, 2017, vol. 2018–Janua, no. August, pp. 10–12.
[53] tcpdump, "Tcpdump/Libpcap public repository," 2018. [Online]. Available: http://www.tcpdump.org/. [Accessed: 25-Apr-2018].



Tarun Kulshrestha is working towards the Ph.D. degree, Department of Computer Science and Engineering, at the Indian Institute of Technology (IIT), Roorkee, India. His research interests include Mobile Crowd Sensing and Computing, Internet of Things, Cloud Computing, Wireless Sensor Networks, and Mobile and Pervasive Computing. He is a member of IEEE.

Divya Saxena received the M.Tech. and PhD degrees in CSE from the IIITM, Gwalior, India and Indian Institute of Technology (IIT) Roorkee, India in 2012 and 2017, respectively. Currently, she is working as a Postdoctoral Fellow, in the Department of Computing, Hong Kong Polytechnic University, Hong Kong. Her research interests include Mobile Crowd Sensing and Computing, Mobile and Pervasive Computing, Future Internet, Networking, IoT and Big Data Analytics. She is a member of ACM and IEEE.

Rajdeep Niyogi is an Associate Professor in the Department of Computer Science and Engineering, Indian Institute of Technology (IIT) Roorkee, India. He did PhD in Computer Science and Engineering from Indian Institute of Technology (IIT) Kharagpur, India, in 2004. His research interests include Distributed Systems, Automated Planning, Multiagent Systems, and Applications of logic and Automata theory. He is a member of ACM.

Jiannong Cao received the MSc and PhD degrees in Computer Science from Washington State University, Pullman, Washington, in 1986 and 1990, respectively. He is currently the chair professor in the Department of Computing, Hong Kong Polytechnic University, Hong Kong. His research interests include parallel and distributed computing, wireless sensing and networks, pervasive and mobile computing, and big data and cloud computing. He is a fellow of the IEEE and ACM distinguished member.