Crowdsensing Simulation Using ns-3

Cristian Tanas(✉) and Jordi Herrera-Joancomartí

Universitat Autònoma de Barcelona, Barcelona, Spain
ctanas@deic.uab.cat, jordi.herrera@uab.cat

Abstract. A crowdsensing network is a sensor network in which sensors are users that sense the environment and send the obtained data using, for instance, their smartphones. The performance of such sensor networks depends heavily on the mobility of the users and their willingness to collaborate. It is hard to obtain a stable set of users to evaluate such kinds of sensor networks and, for that reason, studies of crowdsensing networks are scarce. In this paper, we describe how the ns-3 network simulator can be used to simulate some crowdsensing networks with specific characteristics by using the mobility properties of network nodes together with the wireless interface in ad hoc network mode. We model the identification of network nodes with users of the crowdsensing network and we define how to simulate user sensing capabilities. Finally, we present a simulation example for a specific crowdsensing network where users report incidents in the public rail transport.

1 Introduction

Mobile sensing has become one of the most appealing areas in sensor network research over the last decade as the penetration of mobile devices in our daily life activities reaches unprecedented levels [13].

Indeed, smartphones, besides being sophisticated computing platforms, come along with a complete set of sensing tools (positioning -GPS-, motion -accelerometer-, image -camera-, audio -microphone-, among others), empowering these devices to sense and monitor their surrounding environment. Moreover, smartphone owners should not be ignored since they can also detect distinct properties of their surrounding environment and their human-readable descriptions could complement hardware sensor readings, leading to a new level of sensing, that is *smart sensing*. Relying on users to sense the environment and send the obtained data using their smartphones leads the way to a new generation of sensor network, commonly referred as *crowdsensing networks*, where users become *crowdsensors*.

Therefore, we can leverage the power of collective intelligence, as stated by Cardone et al. [9], together with the sensing capabilities of these smart devices in order to build more powerful and richer sensor networks. Although crowdsensing networks can help overcome many of the limitations of existing proposals in sensor networks, their performance strongly depends on the mobility of the users and their willingness

This work has been partially supported by the Spanish Government through project TIN2010-15764 N-KHRONOUS and the UAB grant PIF 472-01-1/E2010.

to cooperate. In other words, the number of users in a crowdsensing network and their mobility patterns determine unequivocally the area that can be sensed. Also, persuading users to participate in the sensing tasks of a crowdsensing network is highly challenging and, for that reason, only a few crowdsensing network application proposals can be found in the literature and all with an experimental nature. For instance, the MetroSense project [8] offers a network architecture for people-centric sensing that harnesses existing urban infrastructure and human mobility to opportunistically sense environmental data, with applications such as BikeNet [11] or SkiScape [10]. However, none of the applications provide any estimation of the number of users or sensor density required to assure their usefulness.

Simulation may seem a viable solution to analyse the performance of a crowdsensing network application before to launch a real deployment. There are many software-based simulation platforms available, including *ns-2* and *GloMoSim* which are the two most popular ones, that provide a simplified model of the real world and a complete environment to conduct extensive simulations in a wide range of possible scenarios. Even though there are many studies for wireless or mobile ad hoc networks (MANETs) simulation [18], there are no proposals to simulate people-centric sensor networks. Nevertheless, due to the similarities of crowdsensing networks with MANET scenarios, we can still use existing network simulators to analyse the behaviour and performance of specific crowdsensing network applications, and to study the impact that parameters such as sensor density or mobility patterns have on these new kind of environments.

In this paper, we present the *ns-3* network simulation platform focusing on its capabilities of modelling, simulating and analysing the performance of a wide range of crowdsensing networks. We harness the mobility and wireless communication properties of the simulator in order to simulate user sensing capabilities and bring a sense of crowdsensing applications into the simulated world. Furthermore, we introduce a new mobility model for users whose motion behaviour is constrained by the underlying infrastructure of the crowdsensing network and provide a simulation example of a real-world crowdsensing application.

The particular kind of sensing scenarios we bring under discussion in this paper are those where users sense their environment for events that are discrete in time and space, regardless of the events' semantics. That is to say, events occur at precise moments of time and at particular locations, and they can only be detected by the users in the nearby area. Then, the overall goal of the crowdsensing network is to detect and inform of all the events that take place in the area it is supposed to cover. Although this may seem a hard simplification, many smart sensing applications, focusing on the presence or absence of an event and allowing an a posteriori processing of the semantics, fall into this category.

This paper is organized as follows. In Sect. 2, we review some of the most popular network simulation tools. Section 3 presents the architecture of the *ns-3* network simulator and a procedure for crowdsensing network simulation with special emphasis on node mobility. In Sect. 4, we introduce a new mobility model for crowdsensors based on graph theory. A simulation example is provided in Sect. 5. Finally, Sect. 6 concludes the paper.

2 State of the Art

In order to provide an insight of the tools we can use in smart sensing-related research, and without claiming completeness, this section presents some of the most popular simulation platforms, both commercial and open-sourced, that can be used in order to simulate the behaviours of a crowdsensing network.

Most network simulators follow an open source philosophy in order to enable researchers to contribute with new models and protocols to the simulation platform. Two of the most popular network simulators in this category are **NS-2** and **GLOMOSIM**. The former is considered to be the *de facto* simulation tool for networking research, including wireless and ad hoc scenarios [3], as confirmed by Kurkowski *et alter* in [18], while the latter offers a discrete-event simulation platform targeting wireless network systems and suitable for large-scale wireless network simulations [27].

In the open source category we can also find **J-SIM** [1], which provides a distinct architecture built upon the notion of the autonomous component architecture (ACA) [25]. Its design principles mimic those of an integrated circuit where different components are assembled together and interactions between these components occur to simulate specific behaviours. However, if performance is a requirement, a better choice would be **SWANS** [4], which is a scalable wireless network simulator built on top of the JiST simulation engine [5]. A notable characteristic of this simulator is that it can run regular, unmodified Java network applications (web servers, multicast protocols, ...) over the simulated network.

Another open source network simulator that is worth mentioning is **GTNETS**, a discrete-event simulation tool targeting general networking research [21]. The architecture and design of the simulator match those of a real network protocol stack and hardware, and is implemented entirely in C++.

There are also simulation tools that only allow application level protocols in the ISO/OSI Reference Model to be simulated, assuming that an approximate emulation of the lower levels is available, including **DIANEMU** [17] and **JANE** [14, 15].

A more recent simulator is **THE ONE** [16], a Java-based network simulator designed to simulate specific Delay-Tolerant Network protocols and applications [12]. It provides a rich framework for simulating scenarios where node mobility is a critical characteristic.

As for commercial suites for network simulation, **QUALNET** [24], **OPNET** [22], and **OMNET++** [23] are worth mentioning. All of them provide a complete set of user-friendly tools for protocol programming, simulation configuration and data visualization. Although they are commercial solutions, they also support academia and research licences which are free of charge.

Despite the great number of available simulators, the majority of them fail to provide a good, in-depth documentation of their architecture and the procedure required to contribute new models or application protocols.

On the other hand, every crowdsensing application scenario has to be simulated under realistic assumptions, being the mobility of the sensors the most compelling one. In order to simulate that mobility we can use either traces or synthetic models. Traces correspond to motion behaviours observed in real-life systems, while synthetic models aim at providing a realistic behaviour of mobile users motion. Although traces are much more realistic, obtaining them is a challenging task. Therefore, the most used models are synthetic ones [7]. Typically, random-based models are used for performance assessment, such as *Random Walk* or *Random Waypoint* where mobile users are assumed to move towards new locations by randomly selecting a direction and speed, and additionally a pause time, in the case of the Random Waypoint model. Furthermore, the *Gauss-Markov* mobility model is often used in order to avoid the erratic motion behaviour presented by the above models since it uses a parameter that tunes the degree of randomness in the mobility pattern.

Most of the existing proposals in synthetic mobility models are already implemented in the majority of network simulators. Nevertheless, new models can be used by means of mobility trace files which contain the motion behaviour of all the mobile nodes in the simulation employing a specific format. A widely used mobility definition format is the *ns-2* mobility format, which provides a simple set of instructions that can be used to indicate the initial and future locations of mobile nodes, as well as their speed. There are several tools that can generate movement trace files using this format, including BonnMotion [6], SUMO [2] or TraNS [19].

3 ns-3 Overview

ns-3 is a completely renewed version of the discrete-event *ns-2* network simulator, making a huge step forward in organization and performance from its predecessor. Designed for both networking research and education, *ns-3* provides a new, well documented core implemented entirely in C++ adding new models, based on well-known abstractions, including a Wifi link type and several mobility models.

ns-3 is built as a library and its functionalities are divided into separate modules which may be statically or dynamically linked to a C++ main program that defines the simulation topology and conducts the actual simulation. Moreover, it provides Python wrappers to conduct simulations using the Python programming language. Existing models can be more or less heavily modified and even new models can be built from scratch using the C++ programming language in order to enable richer simulations.

The core architecture of *ns-3* emulates the real behaviour of complex network systems, including the ISO/OSI protocol stack. The key abstractions defined by the simulator correspond to the most commonly used concepts in networking, such as nodes, applications, transmission channels, etc. So, the basic unit of interaction throughout the simulation is the *Node* which encapsulates the behaviour of any computing device that connects to a network. Therefore, we can resemble the behaviour of web servers, data storage facilities or even individuals carrying a smartphone by referring to them as Nodes in the simulation. Nodes by themselves perform no actions and have to be assembled in a similar way a manufacturer customizes its devices. Figure 1 depicts the basic model of interaction in *ns-3*. As we can see, each Node runs Applications that define its behaviour, has its own protocol stack following the ISO/OSI model, and has peripheral network interface cards (NICs) installed to enable it to communicate with other Nodes in the same network. Additionally, Nodes may be static or mobile, in which case they have associated a mobility pattern that specifies the motion behaviour of the Node around the simulation area.

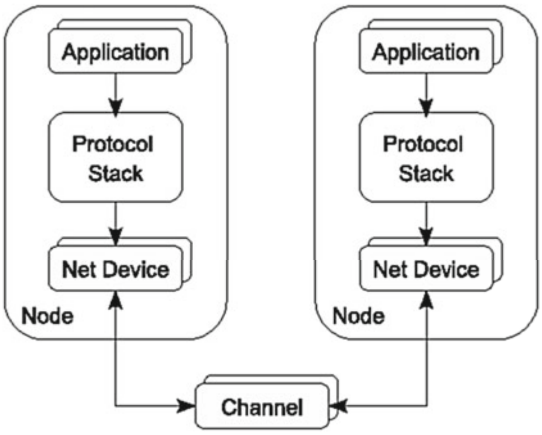

Fig. 1. Basic model of interaction in *ns-3*

3.1 Modelling a Crowdsensing Network

As we have mentioned before, a crowdsensing network is a sensor network where users sense their surrounding environment and send the sensed data using their smartphones. We focus on a specific kind of sensing, that of events that are discrete in time and space. We suppose that a crowdsensing node is able to detect the presence of an event if and only if he or she falls within a given range from the location of the event. Moreover, collective knowledge of the event can be derived from the observations of all the crowdsensing nodes in the area determined by the maximum range at which the event can be detected, considering that users can share information through their smartphones' wireless communication properties.

Given the above assumptions, we can map a user of the crowdsensing network in the real world to a *Node* in the *ns-3* architecture. Then, the simulation will consists of as many nodes as crowdsensors whose behaviour we want to analyse. Every Node in the simulation will have attached a specific *Application* that will emulate the sensing behaviour of the users of a particular crowdsensing network. Therefore, the *Application* will drive the simulation of the interactions between these users.

We can take advantage of the *Node's* wireless communication capabilities in order to enable communication between crowdsensors. In this way, the action of sensing an event and sending the collected information trough the smartphone will be mapped to a network packet sent via a wireless communication channel. Moreover, we can leverage the *ad hoc* mode[1] of a wireless NIC to ensure that the sent data will only reach those nodes that are in the nearby area. The meaning of a "nearby area" can be quantified by restricting the communication range of network Nodes. *ns-3* provides a wireless channel implementation that follows the physical layer model described in [20] and, typically, simulates a wireless channel with a propagation delay equal to the speed of light and a propagation loss based on a distance model, which defines the maximum range at which

[1] All devices are assumed to have equal status in the network and are free to communicate with any other ad hoc device in link range.

a network Node is capable of sending wireless data. Therefore, by restricting a node's maximum communication range we can simulate the maximum distance at which a user of the crowdsensing network could sense the occurrence of an event.

Furthermore, we can take advantage of the mobility properties of a network Node to simulate the mobility of users in a crowdsensing network. A more extensive discussion regarding mobility is provided in the following section.

4 A Graph-Based Mobility Model

Crowdsensing performance depends heavily on the mobility of its users since, for instance, users mobility determines the bounds of the area that the crowdsensing application can cover. Any events, whose location fall outside of this area will remain undetected for the users of the crowdsensing network.

We can benefit from the mobility properties of network nodes in order to simulate the mobility of users in the real world. *ns-3* provides extensive support for user mobility and mobility patterns can be applied to network nodes in different ways. On the one hand, implementation of several well-known mobility models are available for simulation. These include *Random Walk*, *Random Waypoint* or *Gauss-Markov*, among others. Nonetheless, new models can be built from scratch or by combining some of the existing models. On the other hand, mobility models can be specified using the *ns-2* mobility format. A separate mobility trace file must be provided at the configuration phase of the simulation, which the simulator parses in order to set the mobility pattern for each node in the simulation. Note that this method is only valid for those scenarios where the mobility of the crowd is not affected by its sensing activity. In other words, a node can not dynamically alter its destination based on the information sensed from the surrounding environment.

The most used mobility models in wireless or ad hoc network simulation are random-based (e.g. Random Walk, Random Waypoint, etc.). However, the motion behaviour of the users in a crowdsensing application highly differs from such models. Instead, the mobility of users is strongly influenced by their daily activities and by the infrastructure of the area in which they are located. Under these assumptions, we present a graph-based approach to model the mobility of crowdsensing network users, taking into account the time constraints of their daily life activities and those of the infrastructure underlying the sensor network.

We denote by $G = (V, E)$ the graph that represents the underlying infrastructure of a particular crowdsensing network. The set of vertices of the graph, $V = \{v_1, \cdots, v_n\}$, corresponds to the n locations that users may visit, while the set of edges, $E = \{e_{ab} = (v_a, v_b) \quad \text{for} \quad v_a, v_b \in V\}$, map all the possible routes between these locations (e.g. streets or train connections). We also denote by $\omega(e_{ab})$ the weight of the edge connecting vertices v_a and v_b, which represents the relation between the locations mapped to those vertices (e.g. distance, time). Figure 2 depicts an example of such a graph modelling the Barcelona underground service.

Our mobility model is based on paths over the graph G. A path of the graph G is a possible route between two given vertices of the graph, v_{i_1} and v_{i_m}, without revisiting any vertex:

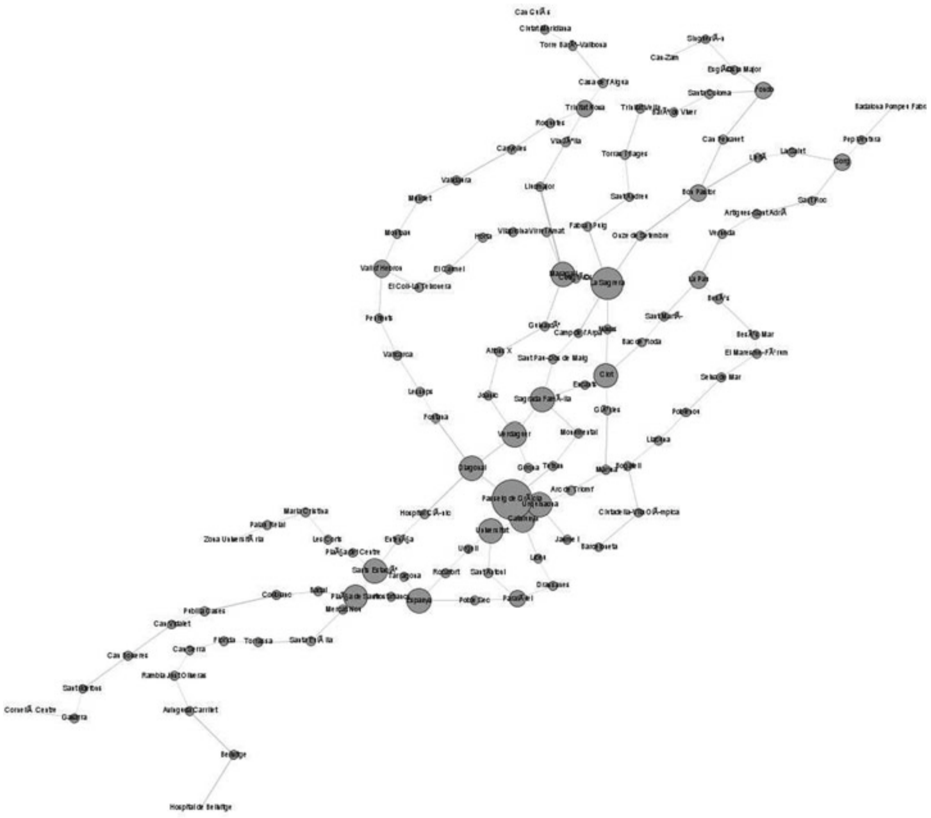

Fig. 2. Example of a graph modelling the underground railway system of the metropolitan area of Barcelona

$$P_{i_1 i_m} = \{(v_{i_1}, \cdots, v_{i_m}) \text{ where } v_{i_j} \in V \ , \ e_{i_j i_{j+1}} \in E\}$$

Since the path can be seen as an edge sequence, a straightforward definition of the cost of a path is the sum of the weights of its edges:

$$\Omega(P_{i_1 i_m}) = \sum_{e_{i_j, i_{j+1}} \in P_{i_1 i_m}} \omega(e_{i_j, i_{j+1}})$$

In order to determine the mobility of the users from vertex v_a to vertex v_b, assuming that more than one path between those nodes may exist in G, we take the one that minimizes $\Omega(P_{ab})$, that is the shortest path. Depending on the definition of the edge weight, the shortest path may minimize time, distance or even other variables such as economic cost.

Besides the path that a user follows to go from one node to another, a simulation environment should define which are the initial and final nodes that have to be assigned to each user of the crowdsensing network. In order to avoid a random selection, we select initial and final destination points for each user through different probabilities.

Every vertex of the graph is assigned two distinct probabilities: an *initial probability*, $\mathcal{P}_{ini}(v_a)$, that expresses the likeliness of the vertex to be selected as initial point, and a *final probability*, $\mathcal{P}_{end}(v_a)$, which determines the likeliness of the vertex to be selected as destination. Then, when selecting the initial and destination points for each node, each vertex is subjected to an independent Bernoulli trial, based on the corresponding probability, which determines whether the vertex is selected as the initial location of the user or as the destination respectively. Notice that the journeys of citizens in a city exhibits such a behaviour as vertices located in the city center are more likely to be selected as destination points, while vertices located in the outer areas of the city are more likely to be the origin of most journeys.

5 Simulation Example

In this section we provide a simulation example of a crowdsensing network where users sense and report incidents in the Barcelona underground service, while they are travelling towards their places of work. Simulations were conducted using the *ns-3.14* version of the *ns-3* simulator, in which we add a new application corresponding to this sensing scenario.

5.1 Simulation Configuration

First of all, we define the simulation area to match the area covered by the underground service of the city of Barcelona. Then, we perform several simulations varying the total number of users (crowdsensors), from 100 to 1200, whose mobility is constrained by the underground infrastructure.

We assume that each user will follow a graph-based mobility model as described in Sect. 4, using a *GraphML* description of the graph seen in Fig. 2. Such graph consists of $|V| = 116$ vertices corresponding to underground stations and $|E| = 262$ edges corresponding to the underground connections between the stations. The weight of each edge, $\omega(e_{ab})$, is defined as the distance (in meters) between the locations represented by vertices v_a and v_b respectively.

The mobility of each user is defined by the shortest path between a pair of stations, probabilistically sampled from the set of all station, following a Poisson sampling model. However, initially, each user will be placed at a distance of 300 m from his or her initial selected station. We use an initial probability distribution based on vertices' degree, assuming that vertices with higher degree are more likely to be selected as destination points as they represent an intersection point of several underground routes. Therefore, vertices having a degree greater or equal to 6 are assigned a final probability $\mathcal{P}_{end}(v_a) = 0.9$, while their initial probability $\mathcal{P}_{ini}(v_a) = 1 - 0.9$. In other words, if 3 or more underground routes intersect in one station, the vertex corresponding to this station will be selected as a destination point with probability 0.9. On the contrary, vertices having a degree less than 6 are assigned a final probability $\mathcal{P}_{end}(v_b) = 0.1$, while their initial probability is $\mathcal{P}_{ini}(v_b) = 1 - 0.1$. This behaviour tries to model a mobility scenario where people living in the outer zones of Barcelona travel by underground to their jobs located mostly in the center of the city.

In order to provide another degree of realism to the simulation, we assume that not all users begin their journeys at the same time, since the timetable restrictions may vary for each user. Therefore, we define an one hour interval for users to begin moving towards their initial assigned station. Then, the final simulation time is determined by the maximum instant of time a user reaches his or her destination.

Incidents, the events that have to be sensed, are generated every 100 s at a randomly-chosen vertex of the graph. Then, we examine all the users whose journeys include the selected vertex between 3 min prior and after the occurrence of the event. From the set of crowdsensors satisfying this requirements, we select one at random which will be responsible for generating a new incident report. Moreover, we demand other users in the area to confirm the generated incident in order to consider it valid. However, we establish a maximum communication range of 100 m. Consequently, only those users located at less than 100 m from the location of the incident will be able to confirm it.

As the users move around the simulation area, the number of crowdsensors available at a specific location will vary with time, defining the node density around the location of the event. Such parameter bounds the maximum number of observations that can be gathered from that event. Moreover, node density is of extreme importance in a crowdsensing network since a value below 1 would mean that most of the events generated would remain undetected.

5.2 Simulation Data Results

Next, we provide the results of our simulation analysis regarding the configuration described in the previous section. In order to avoid biased results due to the standard deviation of the random number distributions used during the simulation, we repeat each simulation 10 times and compute the final results as the average of the results obtained in each individual stage.

First of all, we want to account the total number of incidents that are detected by the users of the crowdsensing network. In Fig. 3 we present the number of detected incidents based on the total number of users of the crowdsensing network. As we can see, the percentage of detected incidents increases from 29 %, in a crowdsensing network formed by 100 users, to nearly 70 % in a crowdsensing network formed by 1200 users. It is worth to notice that even with 100 crowdsensors we can still detect nearly 30 % of the incidents, which is quite acceptable given the large area of the considered scenario. Moreover, since the final simulation time is defined by the maximum instant of time a user reaches his or her destination and the paths that users follow towards their destination may vary significantly in length, the actual number of active crowdsensors at the end of the simulation may fall dramatically.

Using our simulation scenario we can also evaluate the sensor density depending on the total number of sensors for our system. Table 1 illustrates the results, showing the total number of users required to reach a given average user density around the location of an incident. We can observe that the number of users needed grows in a linear way as we increase the required average user density. However, notice that in order to reach a sensor density of 6 around the location of an incident, we only need over 1000 users. That means when an incident is generated there are, on average, 6 users of the crowdsensing network at a distance less than 100 m from the location of the

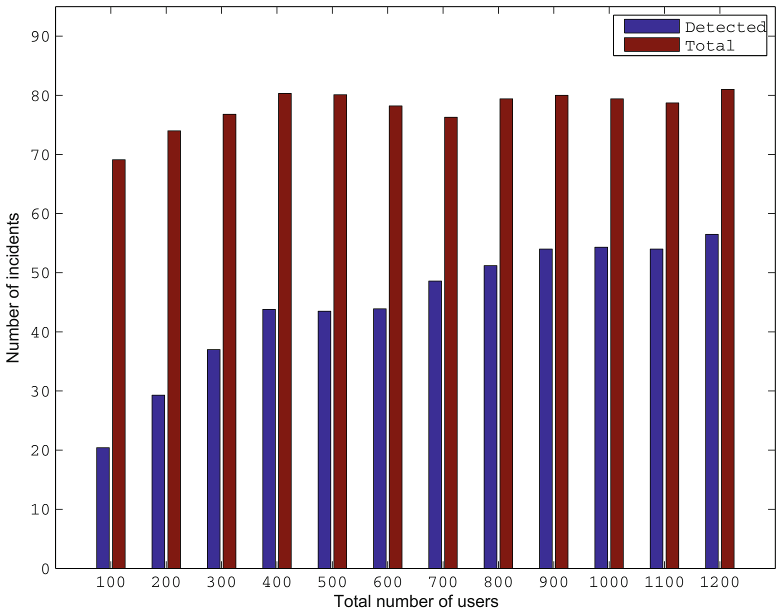

Fig. 3. Number of detected incidents by the users of the crowdsensing network

Table 1. Average node density by total number of users

Total users	100	**200**	300	400	600	800	**1000**	1200
Node density	1.42	**1.91**	2.45	3.15	4.12	5.15	**5.98**	7.01

incident. Therefore, if we request an incident to be confirmed by other users to consider it valid, the sensor density becomes a critical parameter and, in the case of an 1000 users crowdsensing network, we could reach a 5 confirmation threshold. In addition, if our scenario requires just an 1 confirmation threshold for any incident, we could reach that threshold with little over 200 crowdsensors. Although 1000 crowdsensors may seem a large number, we can compare it with the number of users travelling by underground within the metropolitan area of the city of Barcelona supplied by *TMB*, the company that provides the service. In its 2011 annual report [26], *TMB* estimates that nearly 389 million users travelled by underground during that year, which gives us around 1.07 million users travelling by underground in one day (on average). Therefore, 1000 users represent roughly 0.001 % of the estimated underground users on an average day.

6 Conclusion and Further Research

The wide spread and use of smartphones unfolds great potential to effectively map human-centric sensing task to end-user controlled smartphones. However, the performance and usefulness of such sensor networks depends heavily on the mobility characteristics of users and their willingness to cooperate. Moreover, it is hard to deploy a crowdsensing network in a real scenario and validating design principles for crowdsensing networks is a highly challenging task. In this paper, we have presented a software-based approach for crowdsensing network simulation, which allows researchers to analyse the performance of such sensor networks without having to deploy the sensing application on real devices. Furthermore, we propose a graph-based mobility model that takes into account the underlying infrastructure of a crowdsensing network to define mobility patterns for its users. We also provide a simulation example of a crowdsensing network application where users report incidents in the underground service of Barcelona. The data obtained from the simulation shows us that with little over 1000 users, we could detect nearly 70 % of the incidents that occur in the underground system. Nonetheless, predicting the mobility patterns of a crowdsensing network raises complex security and privacy-related issues that we intend to analyse and bring under discussion in further research.

References

1. J-Sim. https://sites.google.com/site/jsimofficial/. Accessed 5 June 2013
2. Simulation of Urban MObility (SUMO). http://sourceforge.net/apps/mediawiki/sumo/index.php?title=Main_Page. Accessed 27 May 2013
3. The Rice University Monarch Project: Mobile Networking Architectures. http://www.monarch.cs.rice.edu/. Accessed 27 May 2013
4. Barr, R.: An efficient, unifying approach to simulation using virtual machines. Ph.D. thesis, Cornell University, May 2004
5. Barr, R., Haas, Z.J., van Renesse, R.: Jist: an efficient approach to simulation using virtual machines. Softw.: Pract. Experience **35**(6), 539–576 (2005). http://dx.doi.org/10.1002/spe.647
6. University of Bonn: BonnMotion. A mobility scenario generation and analysis tool. http://net.cs.uni-bonn.de/wg/cs/applications/bonnmotion/. Accessed 27 May 2013
7. Camp, T., Boleng, J., Davies, V.: A survey of mobility models for ad hoc network research. Wireless Commun. Mob. Comput. **2**(5), 483–502 (2002)
8. Campbell, A., Eisenman, S., Lane, N., Miluzzo, E., Peterson, R.: People-centric urban sensing. In: Proceedings of the 2nd Annual International Workshop on Wireless Internet, WICON '06. ACM, New York (2006)
9. Cardone, G., Foschini, L., Bellavista, P., Corradi, A., Borcea, C., Talasila, M., Curtmola, R.: Fostering participaction in smart cities: a geo-social platform. IEEE Commun. Mag. **51**(6), 112–119 (2013)
10. Eisenman, S., Campbell, A.: SkiScape sensing. In: Proceedings of the 4th International Conference on Embedded Networked Sensor Systems, SenSys '06, pp. 401–402. ACM (2006)
11. Eisenman, S., Miluzzo, E., Lane, N., Peterson, R., Ahn, G.S., Campbell, A.: BikeNet: a mobile sensing system for cyclist experience mapping. ACM Trans. Sen. Netw. **6**(1), 6:1–6:39 (2010)
12. Fall, K.: A delay-tolerant network architecture for challenged internets. In: Proceedings of the 2003 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications, SIGCOMM '03, pp. 27–34. ACM, New York (2003), http://doi.acm.org/10.1145/863955.863960
13. Ganti, R.K., Ye, F., Lei, H.: Mobile: current state and future challenges. IEEE Commun. Mag. **49**(11), 32–39 (2011)

14. Gorgen, D., Frey, H., Hiedels, C.: Jane-the java ad hoc network development environment. In: 40th Annual Simulation Symposium, ANSS '07, pp. 163–176 (2007)
15. Görgen, D., Frey, H., Hiedels, C.: Jane-a simulation platform for ad hoc network applications. In: Demos of the 9th International Symposium on Modeling, Analysis and Simulation of Wireless and Mobile Systems (2006)
16. Keränen, A., Ott, J., Kärkkäinen, T.: The ONE simulator for DTN protocol evaluation. In: SIMUTools '09: Proceedings of the 2nd International Conference on Simulation Tools and Techniques. ICST, New York (2009)
17. Klein, M.: Dianemu - a java based generic simulation environment for distributed protocols. Technical report, Sixth International Workshop on Network-Based Information Systems (NBIS2003), in the framework of the 14th International Conference on Database and Expert Systems Applications 2003 (2003)
18. Kurkowski, S., Camp, T., Colagrosso, M.: Manet simulation studies: the incredibles. SIGMOBILE Mob. Comput. Commun. Rev. **9**(4), 50–61 (2005). http://doi.acm.org/10.1145/1096166.1096174
19. Laboratory for Communications and Applications (LCA): Traffic and Network Simulation Environment (TraNS). http://lca.epfl.ch/projects/trans/. Accessed 27 May 2013
20. Lacage, M., Henderson, T.R.: Yet another network simulator. In: Proceeding from the 2006 Workshop on ns-2: The IP Network Simulator, WNS2 '06. ACM, New York (2006). http://doi.acm.org/10.1145/1190455.1190467
21. Riley, G.F.: The georgia tech network simulator. In: Proceedings of the ACM SIGCOMM Workshop on Models, Methods and Tools for Reproducible Network Research, MoMeTools '03, pp. 5–12. ACM, New York (2003). http://doi.acm.org/10.1145/944773.944775
22. Riverbed Technology: OPNET. http://www.opnet.com. Accessed 7 June 2013
23. Imre, S., Keszei, C.: Simulation environment for ad-hoc networks in omnet++. In: IST 2001 - "Technologies Serving People" (2001)
24. SCALABLE Network Technologies: QualNet. http://web.scalable-networks.com/content/qualnet. Accessed 7 June 2013
25. The Ohio State University and University of Illinois: The Autonomous Component Architecture. http://j-sim.cs.uiuc.edu/whitepapers/aca.html. Accessed 27 June 2013
26. Transports Metropolitans de Barcelona: Resum de gestió 2011. http://www.tmb.cat/ca/c/document_library/get_file?uuid=c5e4a93e-ed5d-4ad6-bf5a-9d6968f6f375&groupId=10168. Accessed 26 June 2013
27. Zeng, X., Bagrodia, R., Gerla, M.: Glomosim: a library for parallel simulation of large-scale wireless networks. In: Proceedings of the Twelfth Workshop on Parallel and Distributed Simulation, PADS 98, pp. 154–161 (1998)