future internet

MDPI

Article

Smart Collection of Real-Time Vehicular Mobility Traces

Nisrine Ibadah[1,*], Khalid Minaoui [1], Mohammed Rziza [1], Mohammed Oumsis [1,2] and César Benavente-Peces [3]

[1] LRIT Laboratory, Associated Unit to CNRST (URAC 29), Rabat IT Center, Faculty of Sciences, Mohammed V University in Rabat, Rabat 1014 RP, Morocco; kminaoui1@yahoo.fr (K.M.); mohammed.rziza@gmail.com (M.R.); oumsis@yahoo.com (M.O.)
[2] High School of Technology, Mohammed V University in Rabat, Sale 11000, Morocco
[3] ETS Ingeniería y Sistemas de Telecomunicación, Universidad Politécnica de Madria, Ctra Valencia, km 7, 28031 Madrid, Spain; cesar.benavente@upm.es
* Correspondence: nisrine.ibadah@gmail.com; Tel.: +212-645695225

Received: 4 July 2018; Accepted: 7 August 2018; Published: 9 August 2018

Abstract: Mobility trace techniques makes possible drawing the behaviors of real-life movement which shape wireless networks mobility whereabouts. In our investigation, several trace mobility models have been collected after the devices' deployment. The main issue of this classical procedure is that it produces uncompleted records due to several unpredictable problems occurring during the deployment phase. In this paper, we propose a new procedure aimed at collecting traces while deployment phase failures are avoided, which improves the reliability of data. The introduced procedure makes possible the complete generation of traces with a minimum amount of damage without the need to recover mobile devices or lose them, as it is the case in previous mobility traces techniques. Based on detecting and correcting all accidental issues in real time, the proposed trace scanning offers a set of relevant information about the vehicle status which was collected during seven months. Furthermore, the proposed procedure could be applied to generate vehicular traces. Likewise, it is suitable to record/generate human and animal traces. The research outcomes demonstrate the effectiveness and robustness of the smart collection algorithm based on the proposed trace mobility model.

Keywords: vehicular trace; mobility traces; Morocco traces; tracking; GPS geolocalization; real-time

1. Introduction

Mobility modeling has been a growing research field since the late 1990s, due to its relevance in the development of modern wireless networks. Thanks to many up-to-date technological advances, diverse trace mobility models have been proposed based on assorted surveillance systems such as Internet of Things (IoT) networks, inexpensive microcomputers, and wireless sensors [1]. Nowadays, tracking systems can generate a set of mobility traces without considering who is tracked and when data are collected [2]. This means that we do not know when our whereabouts are being monitored and tracked using numerous applications [3] that record the personal activities in our daily lives, such as telecoms operators using phones, police radar via car license plates, and public spaces by cameras. These practices happen without revealing clues that public surveillance are occurring, which makes our mobility a curse and a boon. These mobility traces become highly relevant mostly when they reliably reflect real-life movements. They succeed in modeling the complex nature of human motion, wild animals, and heterogeneous vehicles. This category requires more time to collect traces of a nature of the motion. In addition, their deployment is a troublesome task due to many reasons such as:

the high expenses, required storage memories and powerful batteries. Despite this, the scientific community is persistently eager to understand realistic mobility behaviors using mobile networks where they reveal a great deal in order to validate new protocols and applications using tangible data sets (real traces). A comparison between the classical mobility traces (practical) and synthetic mobility models (based on a software program) is depicted in Table 1.

Table 1. Comparison of mobility traces (real deployment) and synthetic model (software program).

Features	Synthetic Mobility Model	Mobility Traces
Scalable	Yes	No
Similarity to real life	Low	High
Time overhead	Small	Large
Complexity	High	Low
Deployment cost	Low	High
Computation overhead	Large	Low

Typically, multiple traces are collected from diverse mobility sources, such as people, animals, and vehicles. They record the movement through a number of deployed devices during specific time periods. Mobile nodes incorporate different device types to collect data sets, such as iMotes, PDAs, and GPS trackers. The information updates are transmitted to the coordinator (or sink) using various network interfaces, like Bluetooth, GPRS, and 802.1, as shown in Table 2. This track is rarely implemented due to several constraints, such as the high deployment cost and the large number of required devices. As in the case of previously collected traces [4], many implementation problems have occurred during deployment, such as:

- hardware resets, which occur very often and which are unpredictable;
- battery consumption of devices during deployment;
- users stealing or not returning the physical devices. Human factors represent the main problem of equipment retrieval after experiments finish.

These issues prevent the complete extraction of a recorded data set, causing serious damage to the reliability of mobility traces. Many recorded parts are lost from the collected model, further impacting synthetic experiments of the transmission and routing between equipment [5]. However, these models have been shown to be the most effective process to properly understand and validate new protocols, traffic applications, and propagation models to result in coherent scenarios with high similarities to real life.

Given the framework described above, in this paper, we propose an efficient collection process that overcomes the issues described and helps to control all of the aforementioned incidents. Moreover, we develop a whole localization framework that permits the collection of a real-time vehicular trace mobility model for seven months (approximately 214 days). Vehicles are localized based on the GPS [6], where their current location can be tracked in real time or at different time instants using a map. This information is transmitted to the main server based on three diverse notification methods according to the gravity of the incident that occurred. Likewise, a set of instant notifications is received when a predefined event is triggered in order to monitor all accidental problems with the aim of correcting them promptly during deployment. This procedure facilitates the management of logistics, preventing the loss or non-recuperation of devices. Moreover, the system can generate detailed reports and statistics regarding a predefined time period with no need to wait for the test time to elapse. Furthermore, it readily extracts the recorded reports as complete traces. Hence, it eventually allows coherent vehicle traces to be obtained as represent Morocco traces.

The remaining part of this paper is organized as follows. Section 2 describes the background and refers to some related works on trace mobility models. Section 3 describes the implementation details based on the proposed localization platform. Section 4 indicates and discusses the different

functionalities outcomes of the Morocco traces. Finally, in Section 5, we present the conclusions and main remarks.

Table 2. Comparative summary of previous trace models and Morocco traces.

Name of Trace	Mobility Type	Number of Nodes	Duration (days)	Device Type	Network
Cambridge 1 [7]	Human	12	5	iMote	Bluetooth
Intel [7]	Human	9	3	iMote	Bluetooth
Toronto [8]	Human	23	16	PDA	Bluetooth
ZebraNet [9]	Animal	100	365	GPS	GPRS
SanFrancisco [10]	Vehicle (Taxi)	500	30	GPS	GPRS
UMassDieselNet [11]	Vehicle (Taxi)	40	60	Ha-Com Open Brick with AP	802.11b
Morocco Traces	**Vehicle (heterogeneous)**	**36**	**210**	**GPS**	**GPRS**

2. Related Works

Researchers were constantly eager to understand realistic mobility behaviors of mobile networks. Diverse trace mobility models were collected over various time periods. In this section, we aim to present some traces that are previously implemented in a brief survey.

Cambridge [7]: In this experiment, groups of users carried small devices iMotes with them for five days; most of them are students at the Computer Lab of Cambridge University, UK. Moreover, a number of fixed nodes are deployed at the most visited places around Cambridge city. The authors anticipate people to visit various locations like pubs, shopping centers, and marketplaces. Only 12 devices are successfully tracked at the first Cambridge trace due to losing some of the experimented iMotes, as well a lot of hardware problems have occurred. Where, these problems were not detected while deployment than analyzing records of retrieved devices. Only 12 devices are successfully tracked at the first Cambridge trace due to losing some of the experimented iMotes, as well a lot of hardware problems have occurred. Where, these problems were not detected while deployment than analyzing records of retrieved devices.

Intel [7]: This data set contains users' Bluetooth sightings in Intel Research Cambridge Corporate Laboratory for about 6 days. They were distributed amongst researchers and staff which were represented as 'interns'. However, residual devices are recognized as 'externals'. 9 iMotes are correctly collected where node 1 was steady and iMotes from 2 to 9 were mobile. But, widely of traces set only 3 days.

Toronto [8]: Researchers gathered Bluetooth traces in various urban environments with the aim of investigating the viability of a large-scale Bluetooth worm outbreak in practice. Devices were carried by students at Toronto University for 16 days. This data set was collected by activating Bluetooth on 23 Palm Tungsten-T PDAs with 16 MB of RAM and a PalmOS to scan Bluetooth devices.

ZebraNET [9]: In this experiment, the recorded data represent the movement traces regarding two real scenarios of ZebraNet deployments at Sweetwaters Game Reserve of Nanyuki, Kenya. The first experiment was in January 2004 and the second was in summer 2005. The hardware of the sensor nodes was mainly composed of a MSP430 processor, flash memory, a radio interface module, and a GPS module. The collected data set used Unix Time-Stamp format and provided detailed information of animals location.

San Francisco Taxi Trace [10]: The records obtained in this test included traces of 500 taxis in the San Francisco Bay area taken for 30 days. Each taxi is equipped with a GPS receiver which indicates the identifier, timestamp, and geo-coordinates. Cab mobility traces were provided by the Exploratorium during the cabspotting project development.

UMassDieselNet [11]: This was also a vehicular DTN trace experiement which was implemented by the UMass Amherst branch of the Pioneer Vallet Transport Authority (PVTA). The collected data represent the daily run of 40 buses on the road. A bus scans other buses 100 times per second which uses an available (discovered) 802.11b radio access point (AP) to access the media and receive incoming

connections while they are in the AP coverage range. Buses in DieselNet were equipped with a desktop computer including an HDD with 40 GB storage and a GPS receiver .

These traces are collected for diverse mobility types, such as humans, animals, and vehicles. Various network interfaces are used in order to transmit information updates, like Bluetooth, GPRS, and 802.1. These record a number of devices during a specific duration where different device types are carried by mobile nodes, such as iMotes, PDAs, and GPS trackers, to collect data sets. This track is rarely applied due to several constraints, as the high deployment cost where a large number of devices are usually expensive. However, this model remains the most effective process to properly understand and validate new protocols, traffic applications, and propagation models. This results in high similarities with real-life outcomes. Meanwhile, these traces are summarized in Table 2. In the next paragraphs, the experiments taken as a reference to compare with our proposal are briefly described.

3. Implementation Details

3.1. Software Implementation Process

The developed framework is based on several realization steps. Each step intervenes to accomplish the main goal, which is to collect real-time mobility traces by controlling all deployment issues. This section presents some of the abundant technical principles on which the internal architecture relies. Furthermore, trackers are incorporated and integrated into vehicles, as shown in Figure 1.

Figure 1. GPS tracker connection into one of our vehicles.

In the implementation, we consider a client-server architecture that uses up-to-date development techniques, as depicted in Figure 2. This combination provides an implementation with full functionality due to the following facts:

- **AngularJS** is based on 'data binding' to prevent dirty-checking loops, which facilitates interface views [12].
- **Node.js** affords real-time updates of framework server without a manual actualizing [13].
- **Nginx** updates of framework clients. In particular, it enhances the fast treatment of node.js. Meanwhile, it resolves the concurrently handling ten thousand connections (C10K) problem, which optimizes network sockets to handle a large number of clients at the same time. This allows flexible state changes to manage several simultaneous connections. Furthermore, it grants a reverse proxy that optimizes the server IP address [14].
- **MVC (Model–View–Controller)** provides a modular framework that separates the view, controller treatment, and connections between the database and client [15].

- **PM2 (Process Manager 2)** allows the server framework to be run continuously even if the framework server is in the offline mode of the node.js [16] in addition to monitoring the server state, as depicted in Figure 3.

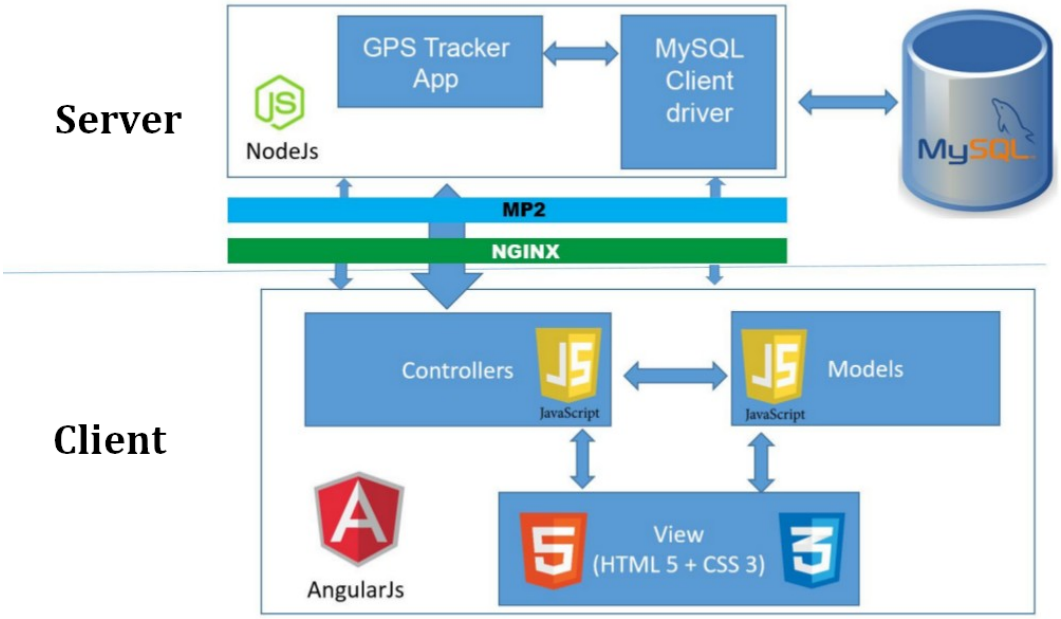


Figure 2. Client/server architecture of the whole tracking framework.

According to Figure 3, we develop two different versions of the framework. This development makes it possible to test the platform that shows a stable version (behavior). This function requires an Apache Subversion (SVN), which allows the current and historical versions of software files of all different concurrent framework developers to be maintained without losing any source code, web page, or documentation. This technique is mainly related to Fredistrano, which automatically exports file sources from a subversion repository and synchronizes them with the content of a target directory. The detailed implementation process of the whole framework is displayed in Figure 4.

```
[[root@vps40349 ~]# pm2 list
● Agent Online | Dashboard Access: https://app.keymetrics.io/#/r/h1it67f809s2yuc | Server name: LWS
```

App name	id	mode	pid	status	restart	uptime	cpu	mem	watching
app.geofleet.ma	2	fork	21994	online	14	2h	0%	72.9 MB	disabled
preprod.geofleet.ma	3	fork	919	online	0	17D	0%	29.2 MB	disabled

Module activated

Module	version	target PID	status	restart	cpu	memory
pm2-logrotate	2.2.0	N/A	online	0	0%	68.688 MB
pm2-server-monit	2.5.1	N/A	online	0	0%	14.859 MB

Figure 3. Server states of the framework.

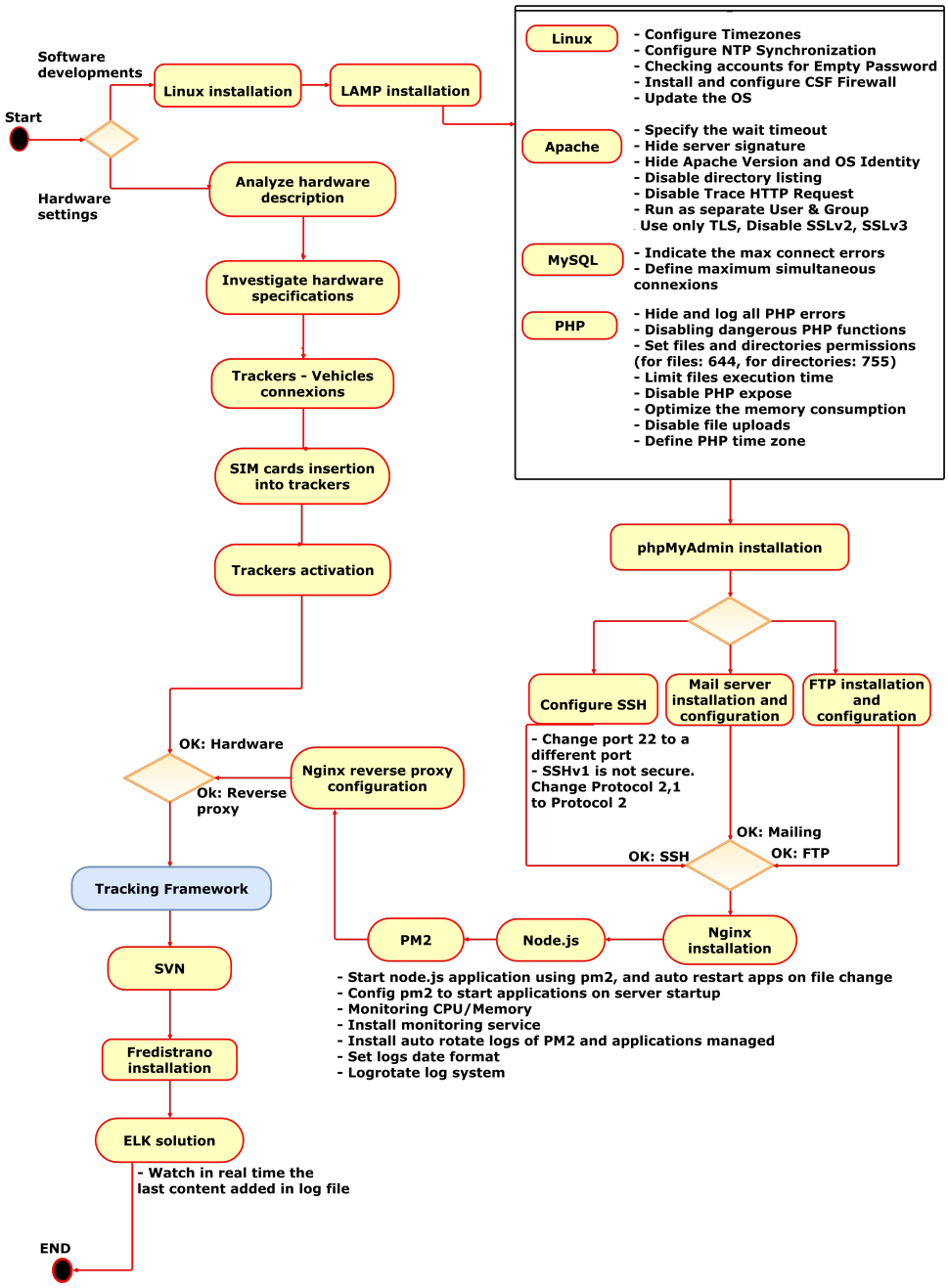


Figure 4. Flow chart describing the processes and procedures required for setting-up the system, its implementation and deployment.

3.2. *Adopted Infrastructure*

Based on the developed framework, the system generates an information which is used either to:

- transmit recorded positions at regular intervals to the geo-localization platform, or
- warn the administrator framework about a system dysfunction.

The system requires network access to transmit the produced updates from trackers to the processing server through any of the available communication networks (GSM and GPRS). The trackers are configured in a hybrid mode; in this mode, they detect an available network to send the collected data immediately, with higher precision and reliability, to the processing center which carries the main framework. These data are analyzed, providing a description of real-life vehicular behaviors.

However, if no network is available in the deployment zone, the tracker stores the collected data until any network is available again and then delivers the data to the server. The chosen operation methods seem to be an appropriate solution given that it offers the major benefit of continuously collecting data during the experiment time without losses. The data set obtained contains a complete start-to-end trace of the vehicle route. The experiments were carried out during a continuous period of seven months (approximately 214 days). It is worth nothing that the adopted approach makes it possible to prevent deployment drawbacks, such as energy consumption, loss of devices, and possible hardware troubleshoots. The chosen solution considers three warning methods to release regulated conditions or triggered updates:

- The main method permits continuous framework updates via GPRS. It allows changes to be displayed at the administrator's dashboard.
- If it is detected that the dashboard is in an inactive state, a warning is forwarded to the administrator by e-mail.
- Otherwise, to carry out permanent vehicle monitoring, these warnings are sent as a text message when specified as critical notifications (optional).

The collection process is displayed in Figure 5. This shows that each vehicle is equipped with a GPS tracker, so that it can be located, and a GPRS modem to transmit current updates. The system computes accurate updates that are delivered to the processing server through an available network in order to provide localization data and all deployment incidents. This allows a smart collection with the aim to detect, correct and prevent losing of any part of mobility model collected. A set of graphs is displayed at the administrator's console in real time. All of the vehicle pathways are recorded in a global data set, making it possible to obtain a new and complete trace mobility model, entitled Morocco traces. The proposed framework monitors vehicle tracks, and additionally it can remotely and concurrently manage the activities of heterogeneous transport fleets in several assorted regions, such as excursions and maritime shipping, as well as carrying out truck localization. Analysis of the overall collected tracks and the current data allows the optimization of courses, for example, in a vehicular ad hoc network (VANET) [17]. Nowadays, these kinds of applications have attracted much attention in the case of smart cities, especially in the deployment of intelligent vehicles, where they can be interconnected in an IoT context [18].

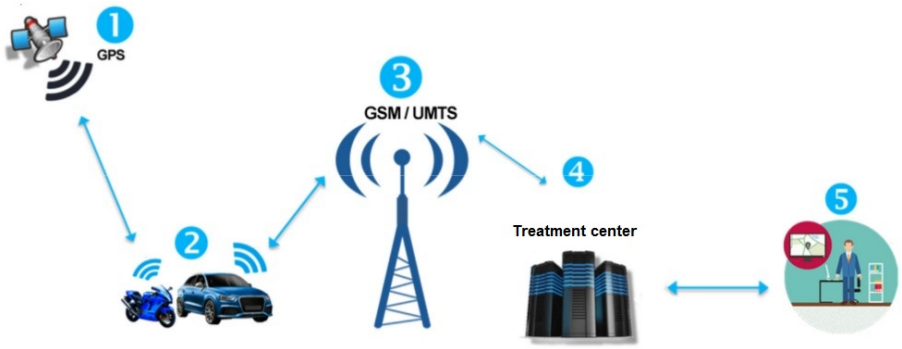


Figure 5. Overview of the system elements and their interactions to perform real-time trace collection.

4. Targeted Goals

This section aims to present some of the main functionalities, as depicted in Figure 6. The implementation makes it possible to obtain complete and coherent outcomes. Based on these functions, we can monitor and administrate all deployed vehicles. This leads to obtaining a smart collection of the final recorded information. Meanwhile, these roles permit a real-time incident detection, which allow for detecting and correcting all deployment issues to further prevent them. This

suggested solution makes it possible to collect an integral traces with any loss thanks to robustness of the developed framework and the important data storage.

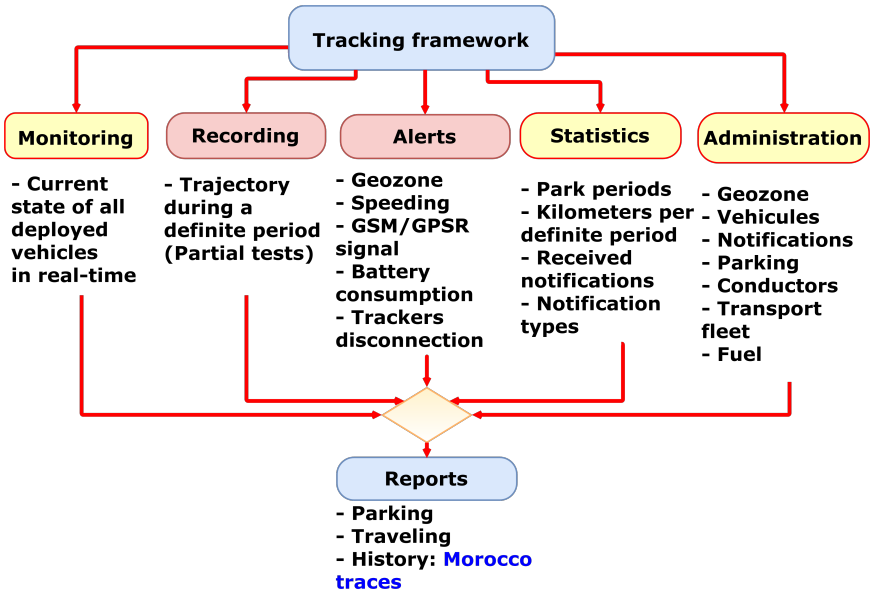


Figure 6. Framework functionalities.

4.1. Monitoring

After the administrator has carried out the authentication successfully, the welcoming tabbed document interface (tab) is Monitoring. It represents the first framework functionality and inspects the GPS trackers' motions by detecting the number of supervised vehicles in each area, the vehicle states, and the corresponding details, as indicated in Figure 7.

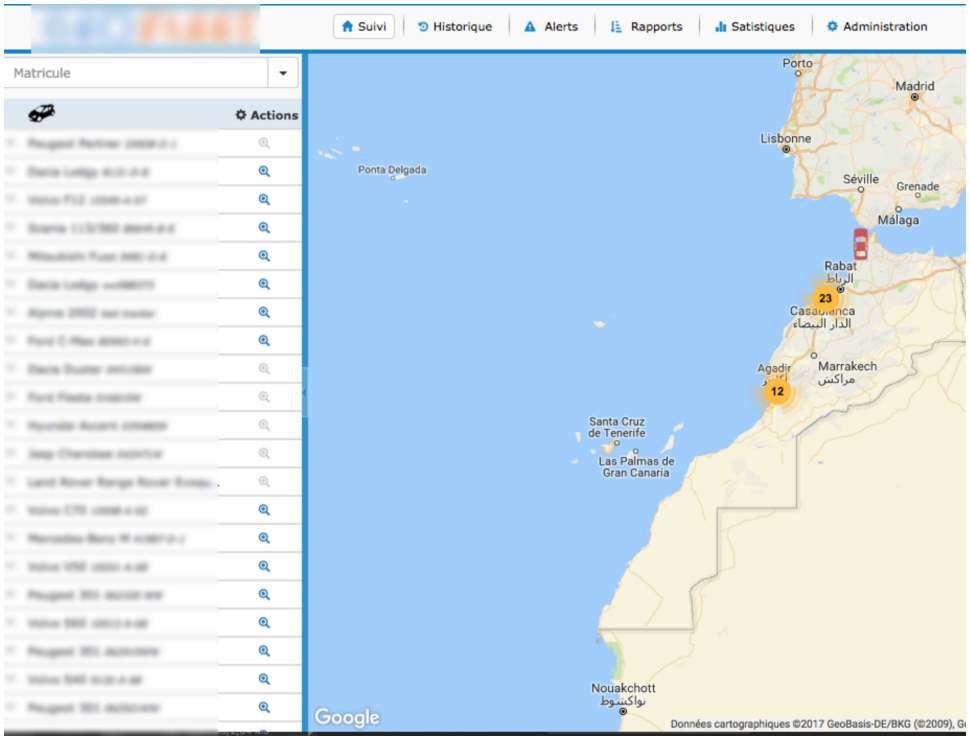


Figure 7. Real-time monitoring tab.

Moreover, we detect deployment problems in real time and we correct framework bugs to prevent the loss of any portion of the data set. For all these vehicles, we install GPS trackers and activate the localization services. Then, we must fill in the corresponding form to add the vehicle information to the platform. These inputs lead to automatic storage in the database of our server, as depicted in Figure 4. Each vehicle has specific information such as the brand, model, year of starting service, registration number, parking times, speed box category, recorded kilometers, and fuel type. Based on these features, the system generates an appropriate warning when a trigger event is set off.

4.2. Recording

The second available functionality is "Recording". This feature is necessary to make us sure about the efficiency of the server, especially when performing instantaneous monitoring. We can simply fill in a time slot of a particular vehicle in order to monitor the correct traveled trajectory, as shown in Figure 8. Moreover, the trajectory recorded in the system reports the real movement of vehicles. We can consider only specific concrete intervals or all of the recorded periods to elicit the appointed reports without stopping the deployed test bench or waiting until the experiment ends. Additionally, global or partial records can be generated, which is one of the required functionalities.

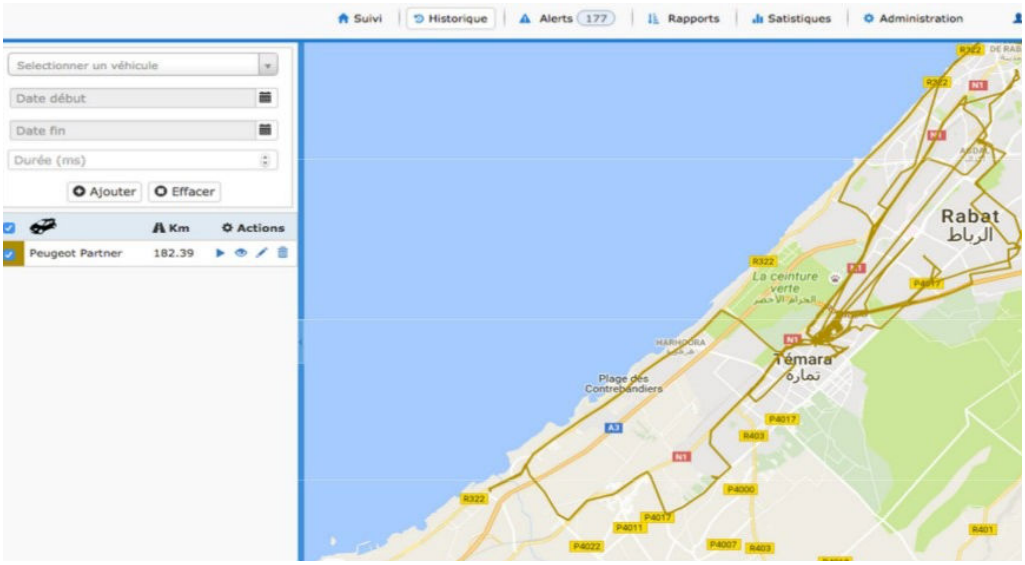


Figure 8. Vehicle trajectory recorded

4.3. Administration

The "Administration" tab is indispensable for providing the real-time platform management process. This window allows the administrator to control all influencing contributors, so we can easily control the efficiency and the worthiness of the whole platform. Diverse administration operations are available to manage the following agents:

- drivers,
- vehicles,
- parking lots,
- customer accounts,
- geo-localization zones,
- notifications, and
- fuel status.

For instance, the case of geo-localization zones is depicted in Figure 9. We restricted a specific vehicle to a certain zone in order to prevent any infraction. When it leaves the assigned area, the administrator is immediately alerted in order to make appropriate decisions. This functionality is useful to monitor and track transport fleets in smart cities context by controlling their pathways to avoid any vehicle leaving the supervised area. The administration sub-tabs harmonize all of the localization tasks with the aim of detecting all deployment problems in real time. At the same time, this smart collection avoids trace losses by a continuous correction of bugs.

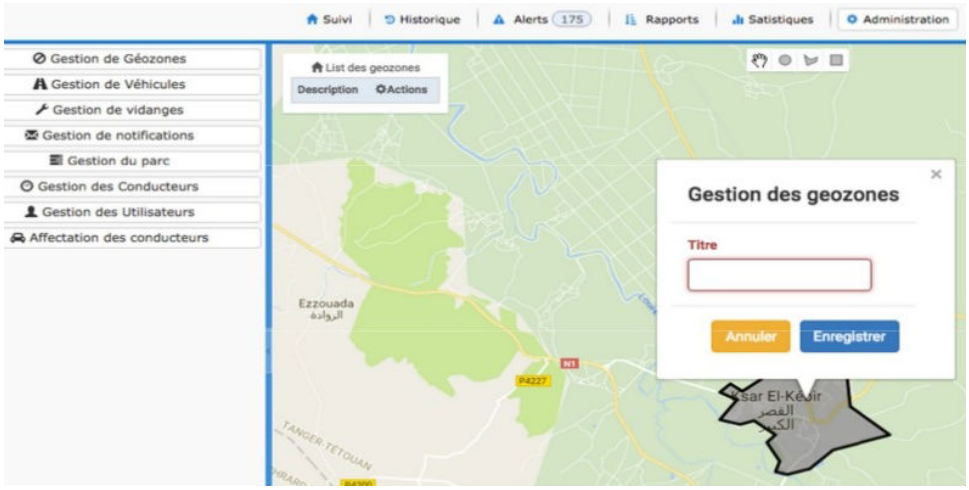


Figure 9. Geo-zone restriction.

4.4. Alerts

The framework administrator receives warning notifications when a predefined event is triggered. These alerts may indicate that the vehicle is entering or leaving the geo-zone, speed changes, oil burning, battery consumption, low fuel level, and GPRS signal reception. All of these possible warnings are shown by default on the dashboard framework in real time. If the platform remains inactive during a predefined period, these alerts will be sent to the administrator by email or by text message (SMS) if the alert that occurred is specified as critical. Moreover, PM2 [19] allows for flexible alert monitoring, as shown in Figure 10.

Figure 10. PM2 alert monitoring.

Figure 11 shows an example where "vehicle speeding" is noticed. When the vehicle exceeds a specific speed limit, the administrator is informed by setting a warning which indicates the recorded speed and the full timing of the event. This operation is performed in real time to warn the vehicle

driver about this penalty infraction on time. This alert contains a collection of useful information including the vehicle, driver, and event details, such as the conductor stamp, exact timing, latitude, longitude, and recorded speed.

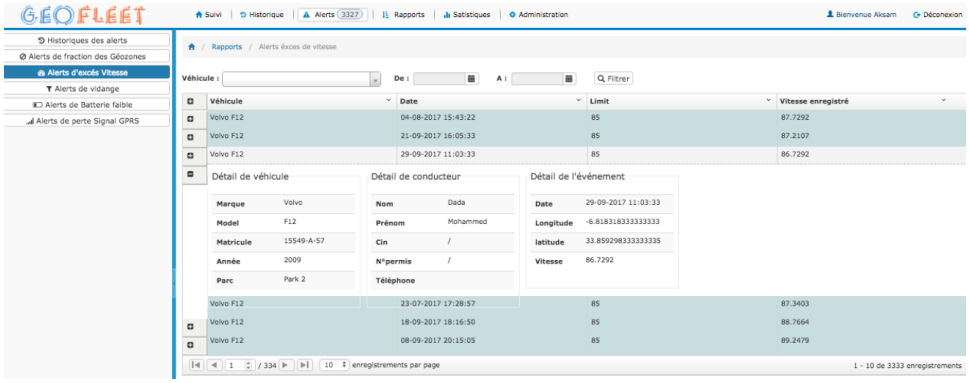


Figure 11. Screenshot of the server console depicting the speeding alert reporting information details.

All of the recorded notifications are classified by type, as shown in Figure 12. It is worth noting that, during the whole deployment and testing time for seven months,

- the absence of the GSM/GPRS signal represents 18% of all notifications,
- reports about battery exhaustion correspond to 9%, and
- tracker disconnection issues identify 5% of reports.

These notifications are generated periodically by default or immediately after an incident happens. These alerts are instantaneously corrected to generate a complete report thanks to the continuous monitoring, recording and administration of the tracking framework. They are drawn on the "statistics tab" depending on the filtering parameters.

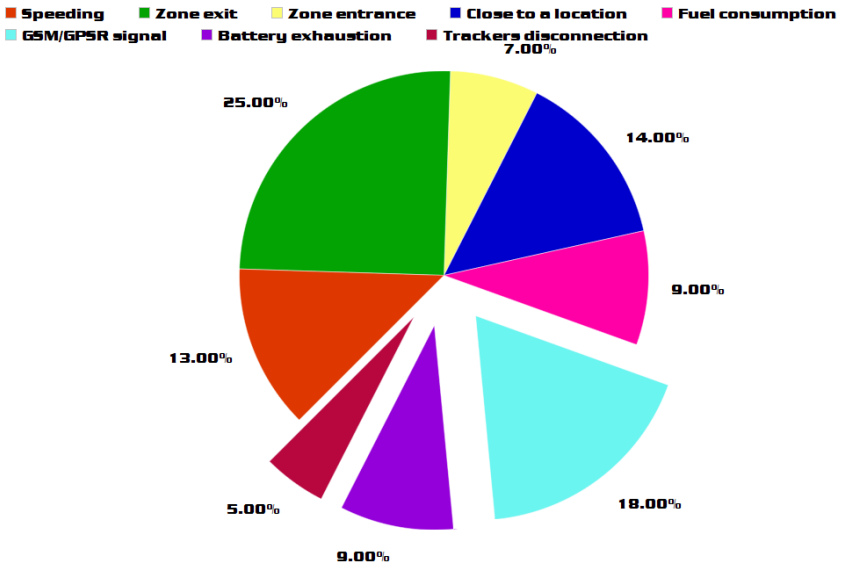


Figure 12. Chart representing the percentage value of the appearance of the different types of notifications.The most frequent is 'zone exit'.

4.5. Statistics

After collecting different real-life reports in the "Statistics" tab, we draw some meaningful graphs during and after the collection of traces. By analyzing these graphs, we can model the movement behaviors of the test vehicles for the whole of the experiment time or only for specific periods without stopping or waiting for the deployment time to elapse, in contrast to previous traces. These results summarize all triggered events that are predefined as alerts in the system. The testing period of the experiment started at the beginning of May 2016 and finished at the end of November 2016. This platform was operating for about seven months (close to 214 continuous days), during which time permanent monitoring, collection, maintenance, and improvement of the whole framework were carried out.

Figure 13 shows some examples of the statistics produced as follows:

- The top left sub-figure indicates the number of speeding notifications received per month by a specific vehicle.
- The bottom left sub-figure presents the exact distance traveled by the defined vehicle during the experiment.
- The top right sub-figure shows the total parking hours per month, indicating that the tracked vehicle accumulated about 175 parking hours in August.
- The bottom right sub-figure shows the distribution of notifications by type that is marked as speeding alerts for the tracked vehicle.

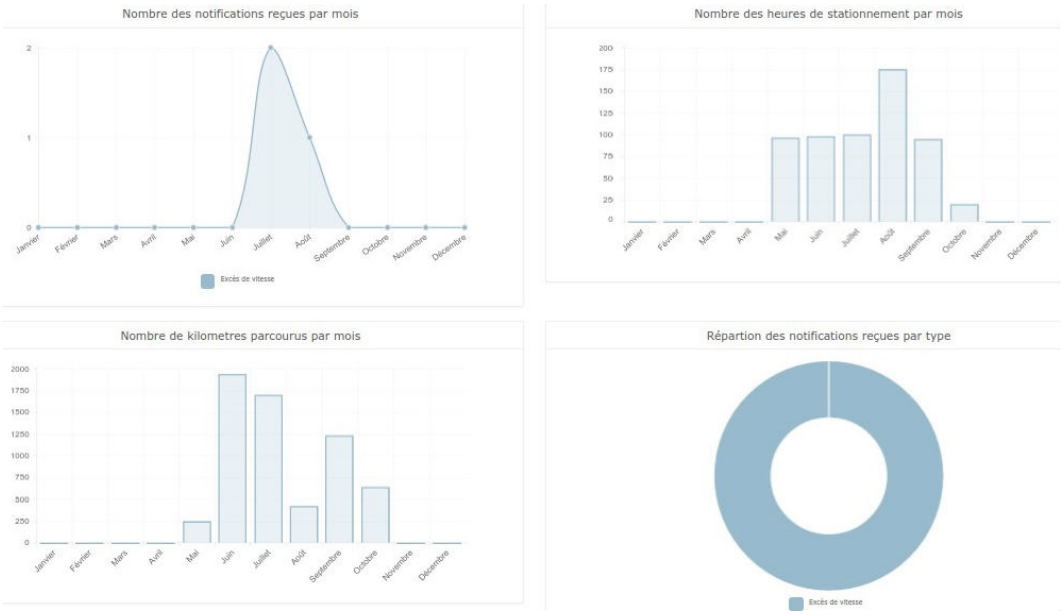


Figure 13. Screenshot of the server console showing the capabilities to draw different types of statistics.

Based on this functionality, we can easily analyze the details of all mobility activities as needed for a specific tracked vehicle or all of them. After collecting the whole data set, we can calculate all previous features in addition to other movement information for all the vehicles or only for a particular user from the global data set collected. Figure 14 highlights the total parking hours of all the vehicles tracked during the experiment. Recording a maximum value means that a low mobility is detected during the whole of the testing period. We note that values recorded remained high during the two first months for all the tracked vehicles. Afterwards, they decreased dramatically in August, reaching 14,040 h, and increased slightly up to 20,789 h in November. This result is inversely proportional to the vehicle's degree of mobility, so it is important to indicate the number of driving hours recorded on mobility reports.

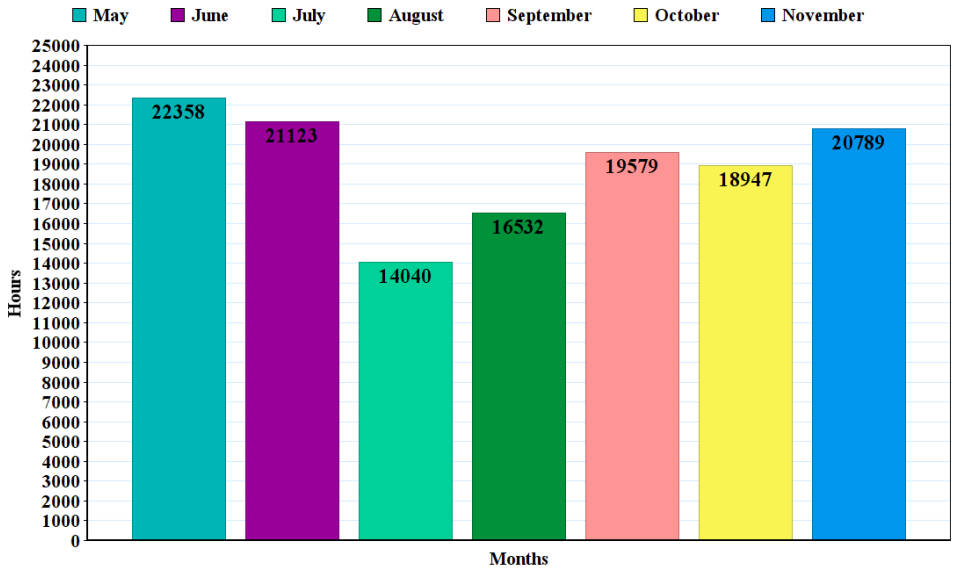
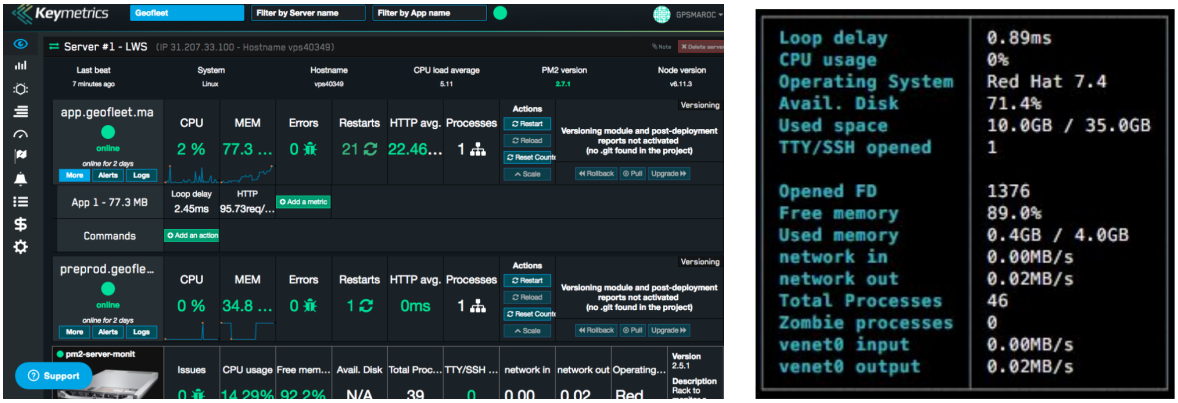


Figure 14. Distribution of the recoded total parking hours per month.

The statistical results demonstrate the robustness and efficiency of our real-time vehicular localization framework to generate a complete trace without losing any part of the data set. Permanent performance monitoring must be implemented to control the system and server operation. These tasks are mandatory to ensure the smoothness of the framework process. Furthermore, using PM2, we control the running framework states in real time, as shown in Figure 15.



(**a**) (**b**)

Figure 15. PM2 real-time framework statistics: (**a**) real-time monitoring; (**b**) framework server details.

Thanks to the detailed information collected, an accurate value is calculated to note the average end-to-end delay between GPS trackers and the administration platform. This duration points out the whole occupied timing from the second to the fifth step in Figure 5. Due to the continuous monitoring, improvement, and analysis of the platform performance during the collection period, we reduced the delay remarkably from 84 seconds in May to 17 seconds in November, as shown in Figure 16. This particularity agrees with the goal of the real-time framework, which is also supported by the successful correction of all detected bugs and problems while experimenting with the software modeling.

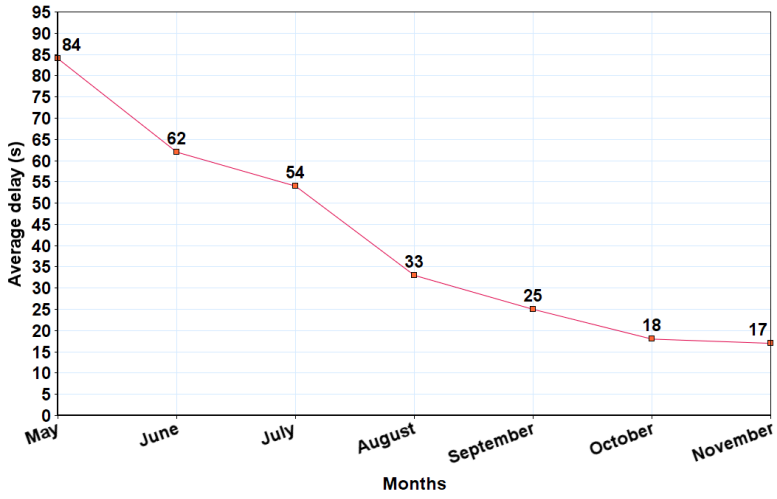


Figure 16. Evolution of the end-to-end delay (average) with time. The more the system is operating, the lower the delay.

All of the previously described framework functionalities are integrated into an additional developed mobile application, as depicted in Figure 17. This represents the mobile version (limited options) of the framework, which is also interconnected to the main treatment server.

This application is installed for an administrator account and also for clients in order to grant the administrator with permanent monitoring of all the triggered events. All of these functions help to supervise vehicle movements in real time. When a problem arises, we correct this bug with the aim of keeping coherent and complete reports without any hardware issues, system resets, or battery consumption issues.

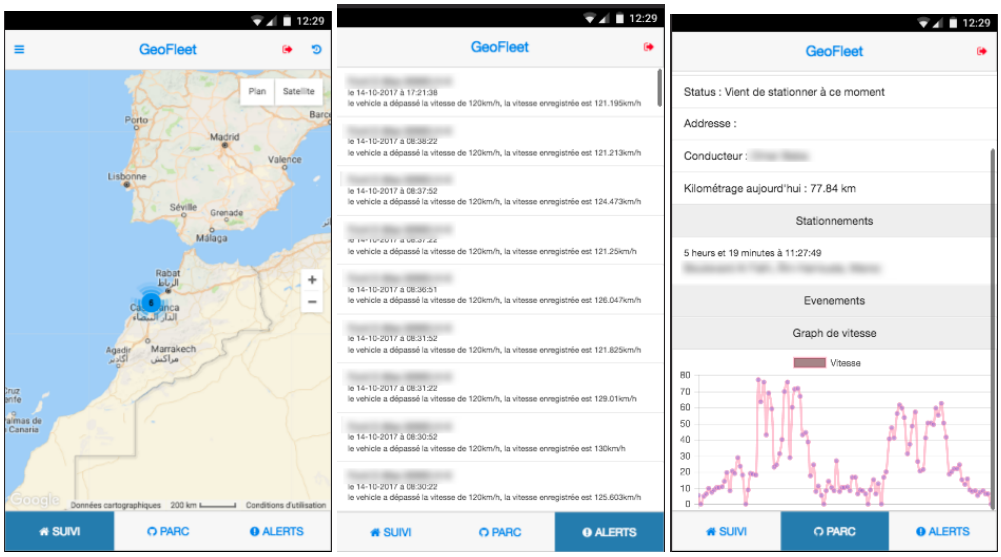


Figure 17. Mobile framework application. Example of simultaneous information provided by the system: traces on map users' information and statistics.

5. Morocco Traces

From previous experiments involving models of mobility traces, they simply distribute a number of devices within a university campus, a conference, or a stadium without any controlling system. Nevertheless, these experiments were used as they provided remarkable feedback about the performance and possible improvements. For example, they noticed the high execution cost and the fact that there was no possibility of restoring devices after suffering a problem. In addition, it was not possible to identify the intended deployment problems, which cannot be detected while the experiment.

For the proposed solution in the case of this paper, problems such as hardware resets, power consumption, and device breakdown are supervised and controlled by the administrator in real time. That is, a set of global information is continuously generated to guarantee the efficiency of the tracking server, as obviously shown in Figure 18.

As mentioned before, our vehicle trace mobility model is entitled Morocco traces and was tested for seven months. We design the developed framework to obtain complete records that exactly reflect the real vehicular motions (real-time), solving any deployment problems (smart collection). Moreover, these traces can be further analyzed to extract a number of mobility features that affect the performance of the mobile network, such as the contact time, activity range, and visiting time. The overall information collected will contribute to obtaining and improving the mobility model. The framework is developed for vehicular mobility usage. However, it is also suitable for any other kinds of targets, such as human and animal motions. A set of reports that stores real traces is collected. Three of them are processed and interpreted at the main processing server to smooth vacant report tabs in our framework. Figure 19 depicts them as follows:

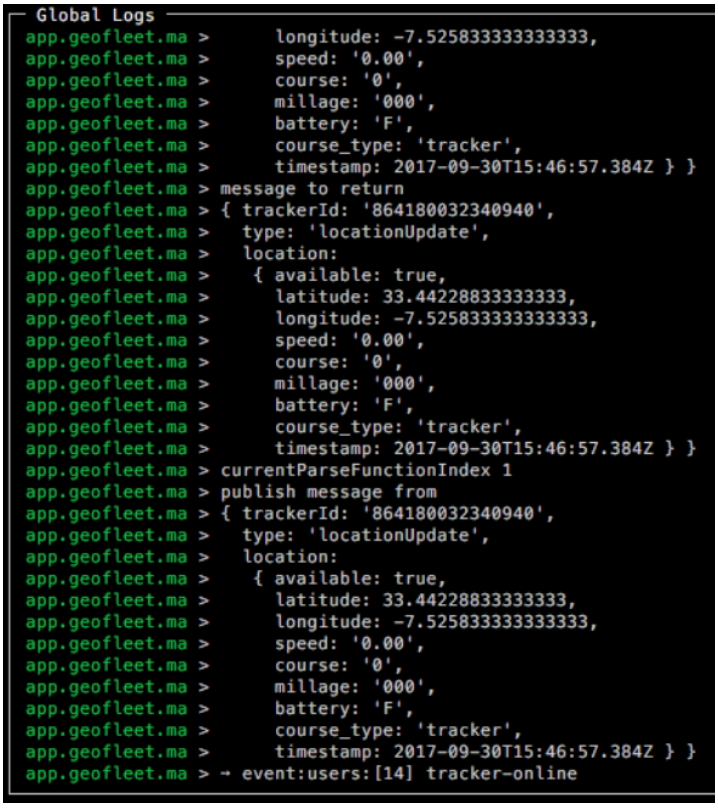


Figure 18. Real-time reports. A dashboard screenshot showing the available real-time reports where the main reported information can be generated.

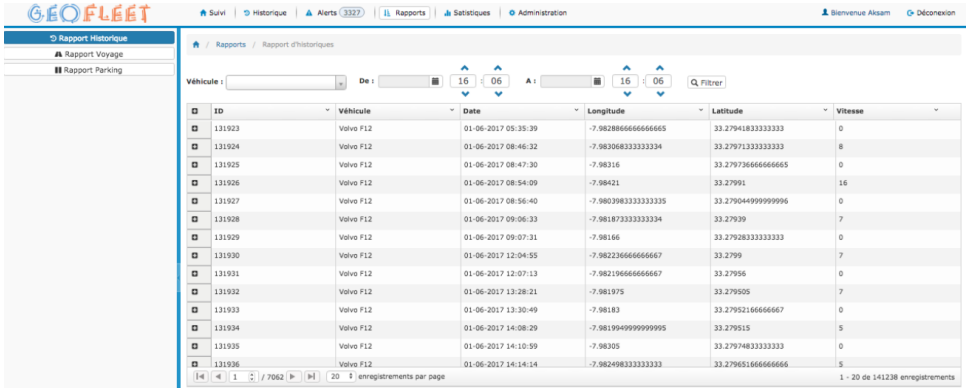


Figure 19. The nerve console showing the history reports of the system performance.

- **History:** This represents the globally targeted traces. These reports show a general overview that represents the main objective and is the heart of our framework. This trace generates exactly a complete trace mobility model. They define a data set that records all registered events during the whole deployment and test period. It also depicts the vehicle identifier, brand, full timing, longitude, latitude, and current speed. We obtain a coherent report with any losses due to detecting, correcting and preventing in advance the most occurred deployed problems. Moreover, all of the developed functionalities are combined in order to achieve and enhance the most effective collection strategy.
- **Traveling:** This is a sub-report of the "Morocco traces" system, which just records the traveling trajectories to and from predefined locations and includes the details of any event.
- **Parking:** This is a sub-trace of "Morocco traces". It produces a log of all the information regarding the vehicle parks.

Traveling and parking reports are obtained to verify the consistency and stability of the collected data set. These reports can be downloaded as text files or visualized as statistics. A comparative summary of the parameter details of some mobility traces and our proposed trace is outlined in Table 2.

In the deployed test, the recorded trace corresponds to 36 vehicles in the first version of "Morocco traces". In the future, we aim to further extend our experiments using as many vehicles as possible, including heterogeneous ones. For this reason, as shown in Figure 20, we have developed a website with the aim of attracting more companies that would allow us to manage their transport fleets. (The website is available at: http://www.gps-maroc.ma/). An important number of mobile devices allows for deep testing and analyzing the server performances in terms of framework synchronization, speed treatment and data storage.

Figure 20. Morocco geo-localization website.

6. Conclusions

Up to today, several trace mobility models have already been developed in which recorded traces' results obtained during testing phases are just extracted from the returned devices. This classical deployment strategy has several drawbacks such as equipment loss, unpredictable hardware resets, non-supervision of battery remaining power, and a large variety of problems. In this paper, we propose a novel trace collection approach that avoids the aforementioned deployment issues. Hence, we have designed and developed a complete localization environment operating in real time, which is connected to a GPS aimed at generating tracks. The deployed system provides several functionalities for monitoring, recording, alert management, reporting, and generation of statistics. This platform can be deployed for different purposes such as human mobility, animal surveillance, and vehicle localization. The system was deployed and tested during seven months, resulting in the set of results, so-called 'Morocco traces', which provided real-time traces based on a smart collecting data process. Future research activities will focus on improving the system by adding new languages, such as multilingual capabilities, and increasing the number of simultaneous vehicle traces by extending its use to thousands of vehicles, especially transport companies to efficiently manage their fleets. This activity is aimed at promoting Casablanca as the first smart city in Africa that collects human traces.

Author Contributions: N.I. developed the proposed idea, implemented it, and obtained the results. K.M. and C.B.P. discussed the methodology, supervised the framework development, and reviewed paper. M.R. and M.O. validated the results.

Conflicts of Interest: The authors declare no conflict of interest.

Abbreviations

The following abbreviations are used in this manuscript:

IoT	Internet of Things
GPS	Global Positioning System
GPRS	General Packet Radio Service
DTN	Delay-Tolerant networking
SVN	Apache Subversion
VANET	Vehicular Ad Hoc Network
C10K	Concurrently Handling 10,000 Connections
MVC	Model–View–Controller
PM2	Process Manager 2

References

1. Yeo, H.S.; Lee, B.G.; Lim, H. Hand tracking and gesture recognition system for human-computer interaction using low-cost hardware. *Multimed. Tool. Appl.* **2015**, *74*, 2687–2715. [CrossRef]
2. Pirozmand, P.; Wu, G.; Jedari, B.; Xia, F. Human mobility in opportunistic networks: Characteristics, models and prediction methods. *J. Netw. Comput. Appl.* **2014**, *42*, 45–58. [CrossRef]
3. Tomasello, M. *A Natural History of Human Thinking*; Harvard University Press: Cambridge, UK, 2014.
4. Batabyal, S.; Bhaumik, P. Mobility models, traces and impact of mobility on opportunistic routing algorithms: A survey. *IEEE Commun. Surv. Tutor.* **2015**, *17*, 1679–1707. [CrossRef]
5. Rizzoli, A.E.; Rudel, R.; Förster, A.; Corani, G.; Cellina, F.; Pampuri, L.; Guidi, R.; Baldassari, A. Investigating mobility styles using smartphones: Advantages and limitations according to a field study in Southern Switzerland. In Proceedings of the 7th International Congress on Environmental Modelling and Software, San Diego, CA, USA, 15–19 June 2014; pp. 274–282.
6. Amini, A.; Vaghefi, R.M.; Jesus, M.; Buehrer, R.M. Improving GPS-based vehicle positioning for intelligent transportation systems. In Proceedings of the 2014 IEEE Intelligent Vehicles Symposium Proceedings. Dearborn, MI, USA, 8–11 June 2014; pp. 1023–1029.
7. Chaintreau, A.; Hui, P.; Crowcroft, J.; Diot, C.; Gass, R.; Scott, J. Impact of human mobility on opportunistic forwarding algorithms. *IEEE Trans. Mob. Comput.* **2007**, *6*, 606–620. [CrossRef]

8. Su, J.; Chan, K.K.; Miklas, A.G.; Po, K.; Akhavan, A.; Saroiu, S.; de Lara, E.; Goel, A. A preliminary investigation of worm infections in a bluetooth environment. In Proceedings of the 4th ACM Workshop on Recurring Malcode, New York, NY, USA, 30 October–3 November 2006; pp. 9–16.
9. Juang, P.; Oki, H.; Wang, Y.; Martonosi, M.; Peh, L.S.; Rubenstein, D. Energy-efficient computing for wildlife tracking: Design tradeoffs and early experiences with ZebraNet. *ACM SIGARCH Comput. Archit. News.* **2002**, *30*, 96–107. [CrossRef]
10. Piorkowski, M.; Sarafijanovic-Djukic, N.; Grossglauser, M. A parsimonious model of mobile partitioned networks with clustering. In Proceedings of the 2009 First International Communication Systems and Networks and Workshops, Bangalore, India, 5–10 January 2009; pp. 1–10.
11. Zhang, X.; Kurose, J.; Levine, B.N.; Towsley, D.; Zhang, H. Study of a bus-based disruption-tolerant network: mobility modeling and impact on routing. In Proceedings of the 13th Annual ACM International Conference on Mobile Computing and Networking, New York, NY, USA, 9–14 September 2007; pp. 195–206.
12. Braun, E.; Radkohl, A.; Schmitt, C.; Schlachter, T.; Düpmeier, C. A lightweight web components framework for accessing generic data services in environmental information systems. In *From Science to Society*; Springer: Cham, Switzerland, 2017; pp. 191–201.
13. Chaniotis, I.K.; Kyriakou, K.I.D.; Tselikas, N.D. Is Node. js a viable option for building modern web applications? A performance evaluation study. *Computing* **2015**, *97*, 1023–1044. [CrossRef]
14. Gueye, A.D.; Faye, P.M.D.; Lishou, C. Optimization of Practical Work for Programming Courses in the Context of Distance Education. In *Online Engineering & Internet of Things*; Springer: Cham, Switzerland, 2018; pp. 764–777.
15. Reagan, R. Other Tips and Tricks. In *Web Applications on Azure*; Springer: Cham, Switzerland, 2018; pp. 381–414.
16. Shankar, A.R. Essential Game Developer Toolkit. In *Pro HTML5 Games*; Springer: Cham, Switzerland, 2017; pp. 409–420.
17. Wang, H.; Liu, R.P.; Ni, W.; Chen, W.; Collings, I.B. VANET modeling and clustering design under practical traffic, channel and mobility conditions. *IEEE Trans. Commun.* **2015**, *63*, 870–881. [CrossRef]
18. He, W.; Yan, G.; Da Xu, L. Developing vehicular data cloud services in the IoT environment. *IEEE Trans. Ind. Inf.* **2014**, *10*, 1587–1595. [CrossRef]
19. Chiang, C.T. Design of a High-Sensitivity Ambient Particulate Matter 2.5 Particle Detector for Personal Exposure Monitoring Devices. *IEEE Sens. J.* **2018**, *18*, 165–169. [CrossRef]