ICIT 2015 The 7th International Conference on Information Technology
doi:10.15849/icit.2015.0089

VeSimulator A Location-Based Vehicle Simulator Model for IoT Applications

Osama Oransa
Arab Academy for Science, Technology and Maritime Transport
Cairo, Egypt
osama_oransa@hotmail.com

Mostafa Abdel-Azim
College of Computing & Information Technology
Arab Academy for Science, Technology and Maritime Transport
Cairo, Egypt
melbakary@aast.edu

***Abstract*— Location-based services (LBS) are important aspects in today business models where a lot of use cases are built around the identification of user location such as the advertisements, asset tracking, geo-fence, and a lot of different things that utilize such location information. With the maturity of Global Positioning System (GPS) technology many different devices are shipped with built-in GPS unit, this enables the Internet of Things (IoT) application to utilize this location information and build many location-based business models including health, transportation, marketing and social services. Smart transportation is one of these important IoT applications in which LBS play a major role. In this research we built a location-based simulator model that can be used in developing internet of things applications. The research focused on developing a generic simulator model that can be customized according to the used application, so we can develop IoT location-based application without the need to have hardware components in early research phases. We built a train simulator and tested it against a railway control system. This simulator achieved good results in simulating the behavior of a moving train using different testing scenarios.**

Keywords— Internet of things; location-based services; vehicle simulator model; train simulator.

I. INTRODUCTION

Internet of things starts to affect every life aspect and many applications have been implemented in the past few years that change people lifestyle. If we categorize the existing devices into i) connected-devices; where devices has a way to connect and ii) non-connected devices; where devices has no connectivity. IoT enables more devices to be transformed from non-connected into connected world.

The concept is wide-spread to cover not only devices but also solid objects by injecting the required sensors into these objects, for example building a system to monitor the railway bridges to ensure early detection of any bridge damages; this can be achieved by injecting some sensors in the bridge body [1]. The basic concept is the same in all IoT applications; converting things into connected objects that have unique identifiers and are responsible for sharing information or executing an action or both [2].

This enables the efficient utilization of these devices and opens the door for more business applications and better human-machine interactions. If we picked for example the location-based services, they enable the location tracking of different things that can vary from human beings to vehicles. Therefore we can build different business models around such information e.g. location-based marketing [3].

Other usages such as asset tracking, geo-fencing [4] have become essential parts of car security and traffic monitoring. The concept spread to cover additional areas such as using business intelligent analysis to identify traffic status using the collected data from different tracked vehicles and their own speed in different roads.

To develop business intelligent location-based applications a need to have vehicle simulators that facilitate the development of these applications and testing the business model around the proposed services without the need to invest in the hardware until a complete and mature model become clear, this simulator role is essential before building a new LBS system or altering an existing system to reduce the possibility of failing to fulfill system specifications and also to optimize the system performance [5].

doi:10.15849/icit.2015.0089

The structure of this paper is as following; in this section we provide background information of the current location-based IoT services, in the next section we will describe the problem definition, after that we will move to discuss the proposed model for building a location-based simulator, we will then discuss the methods and experimental results and we will finish the paper with research conclusion.

II. Problem Definition

The IoT application has become one of the hottest research areas over the past few years and looking at the location-based services many applications has been proposed to solve existing business problems such as asset tracking, geo-fencing, traffic analysis etc. With more involvement of intelligent analysis of collected location data a requirement to have location based simulators that enable the evolvement of these services using pure software model has become clear.

This can push the creation of more innovative applications that utilize the data generated by the vehicle simulators. One example is developing a railway control system using the location-based services and other services. To simulate such railway control system a requirement to deploy a hardware device to communicate with this control system in different trains to collect the train's location data. This will not only cost us a lot in terms of devices and connectivity in the early stages of the research but it will also slow down the research progress by the hardware capability constraints. One additional challenge here is the need to upgrade of firmware in these devices with each change in the system in particular in the communication protocol.

All these reasons can point to a clear requirement to develop a location-based simulator so we can speed up the research and reduce the time to market in LBS and therefore reduce the overall solution cost. It also enables the flexibility of developing the communication protocol, hardware and device features, etc. The purpose of this research is to develop a generic customizable location-based simulator model that can be used in developing location-based services and in particular for train location tracking.

III. Existing Simulation Models

The existence of many location services in the internet world such as Google Maps® and the inclusion of these services in the smart phone world expanded the application domain of LBS. Different simulators exist to simulate these services with different level of simulator maturity that varies from a simple location output to a very advanced street-viewer simulator as in Google maps street viewer.

If we focus on train simulators as an example, many simulators have been developed using different concepts, in the early stages of these simulators, the simulator has to contain the collection of track geometry and related speed restrictions [6]. Because train simulators are much more complex awareness of track signaling status is also required, so if we need to build a train simulator we need to consider having a messaging system to send the signal details to the simulator to reflect the received railway signals in its behavior.

Two simulation models are usually used; time and event-based models, in time-based models; the time is divided into spaced intervals where movement is evaluated at each interval. This is near real simulation model and easier for development but it needs much more computational power which can be reduced by increasing the time intervals.

While in event-based model, the movement calculations occur only in pre-defined events (e.g. train leave or arrive to station) which lead to less computation but inconsistent movement updates, this ideally fits in timetable or traffic control applications [6].

Many train simulators exist and can be used to do railway simulation if we pick one example as Train Operation Model (TOM) which contains three different type of simulation; Train Performance Simulators (TPS) which simulate a single train movement, Train Movement Simulator (TMS) which simulates the performance of multi-train network and Electric Network Simulator (ENS) that simulate power flow in the railway system [7]. This TOM does the simulation in respect to railway system as a whole not only the train as abstract, another simulation model called TrainSim which is limited to calculating the train speed profiles using different speed calculation methods [8], some more advanced 3D simulators exist as Open Rail train simulator; which is powerful train simulator but doesn't fit the purpose of this research which is providing simulator for IoT applications [9].

In our research we used the time-based simulation technique with a configurable time slicing that can be tailored according to the application nature and we build a model around this simulation technique that also supports the bidirectional communications (inbound and outbound) to allow complete interaction with any LBS system.

IV. Building VeSimulator Model

In order to build the simulation model many steps are required; the first step is data collection. Data collection is the most important step as it collects the data that will be fed into the simulator so it uses it to simulate the vehicle movement. The data should be gathered according to the required business model. In our research we have selected train location data as our source to build a train simulator. The following steps describe what we did to develop our location-based simulator:

A. Gathering vehicle location dataset

In this step we have collected the railway location dataset by using a GPS-enabled device while the train is moving from the start station to the destination station. The device logs the location periodically and allows the user to mark certain location such as starting station, crossing areas, middle stations, and final station locations manually. To generate the data for our research we developed an Android® application to get the train locations and log them into a device local file. Most of smart phones have already Assisted-GPS (A-GPS) technology which gives a higher GPS location accuracy by integrating both mobile network and GPS. The add value from A-GPS is that it provides a quick location fix and a better coverage especially inside buildings [10].

doi:10.15849/icit.2015.0089

In Android OS, there is already support for location APIs that we can use. The main class here is the LocationManager class which is responsible for returning the current device location [11]:

```
LocationManager  locationManager = (LocationManager)
getSystemService(Context.LOCATION_SERVICE);
```

After initiating this LocationManager class, we can get the location by calling getLastKnownLocation() method to get the location.

```
Location location = locationManager
.getLastKnownLocation(LocationManager.GPS_PROVIDER);
```

We can also get the location with each change in device location by registering the application main class to listen to these location changes. In the application we have configured the location updates to be with each 100 meters in distance; this can be configured according to the business model and the required location frequency.

```
locationManager.requestLocationUpdates(
        LocationManager.GPS_PROVIDER,
        10000,   // 10 seconds
        100,     // 100 meters,
        this);   // the listener class
```

It is important to define this location change sampling according to our application requirements for example every 50 meters, 100 meters, or 500 meters. In this code we have configured the location update to be fired every 10 seconds or 100 meters difference from the previous location. With each device location change the onLocationChanged(Location) method gets called with the new device location where we need to implement the location logging logic (in a local device file).

```
public void onLocationChanged(Location location) {
        double latitude = location.getLatitude();
        double longitude = location.getLongitude();
        float accuracy = location.getAccuracy();
        // logging logic here …
}
```

To run this Android application, it will need the permission to access the fine-tuned location of the device as following:

```
<uses-permission
android:name="android.permission.ACCESS_FINE_LOCATION" />
```

Table I contains a sample of collected location data formatted in a table.

TABLE I. SAMPLE OF COLLECTED DATA

Latitude (D)	Longitude (D)	Accuracy	Point type
31.259905	32.300598	30	S
31.259428	32.300255	45	
31.258933	32.299912	30	
31.258492	32.299590	30	
31.257979	32.299246	30	
30.604054	32.300908	30	C
30.603149	32.299878	30	
30.601930	32.298483	45	S

The file as we can see logs the latitude, longitude and accuracy of each location point, the recorded accuracy is very important as it describes the pseudo-range of the reported location, this value depends on many factors such as atmospheric conditions, and GPS device receiver quality, as we will see later this value is important and shouldn't be neglected in most of location-based applications.

The application logs each point with a tag (S, C or nothing) by allowing manual marking of the current location position by either no-mark for normal position and 2 additional mark types either C or S. In our simulator we used C for railway crossing location and S for railway stations. All the logged entries by default are logged without any flag but if the user clicks on station or crossing buttons in the application, the application logs the corresponding flag. In our service we didn't record the altitude of the location as it is not required in our railway system but we can also record it in the collected data if required.

If we are tracing a bus for example the values would be different as we will map the points into bus stations, traffic lights (instead of crossings), squares, etc.

B. Manipulating the location datasets

In this step we manipulate the application output data file to give some meaningful values to the collected data. For instance we can add friendly name to railway stations, update stations flag to distinguish between start station, middle station and final station, add the available number of platforms for each station, and add the max speed allowed for each crossing.

The following three sample lines show the file format after adding such information:

31.259905, 32.300598, 30, SS, Railway Station Name, 3

31.241799, 32.298023, 30, C, 40

30.601930, 32.298483, 45, S, Middle Station Name, 2

…

The first line describes the station as start station (SS) with 3 platforms, the second line describes the crossing location with maximum speed allowance as 40 Km/Hr, and in the third line we have added middle station name and number of platforms in this station.

doi:10.15849/icit.2015.0089

Again if we are building simulator for other applications such as bus simulator, the points should always correspond to our simulation milestones with different meanings:

31.259905, 32.300598, 30, SS, City Central Bus Station, 8

31.241799, 32.298023, 30, C

30.601930, 32.298483, 45, S, Town Centre Bus Station, 2

In the first line we have added the start station name and number of bus parking slots as 8. In the third line we have added the station name and number of minutes the bus will wait there (2 minutes).

C. Building the simulator data model

Building the simulation data model is a challenging step to ensure that the system is flexible for future changes especially to add additional relations in the future.

The model is composed from different inter-related tables where the location table represents the core part and gets connected to other tables that represent the purpose and the usage of our simulator. In our railway model the location dataset is linked to railway data, which is represented by a railway database table which in turn get connected to the following tables; stations, crossings, switches, maps, milestones and events. Milestones represents all the location points that are collected in that railway, these points are what the simulator will use to simulate the train movements using vehicle speed to check-in different points.

In fig. 1 we can see part of the simulation data model that we used in our railway application, the location table is in the heart of the application data model and get connected to almost all major tables in this model, we have created a simple application to import the collected data into our system model; during importing the model, the distance between each location milestone is calculated.

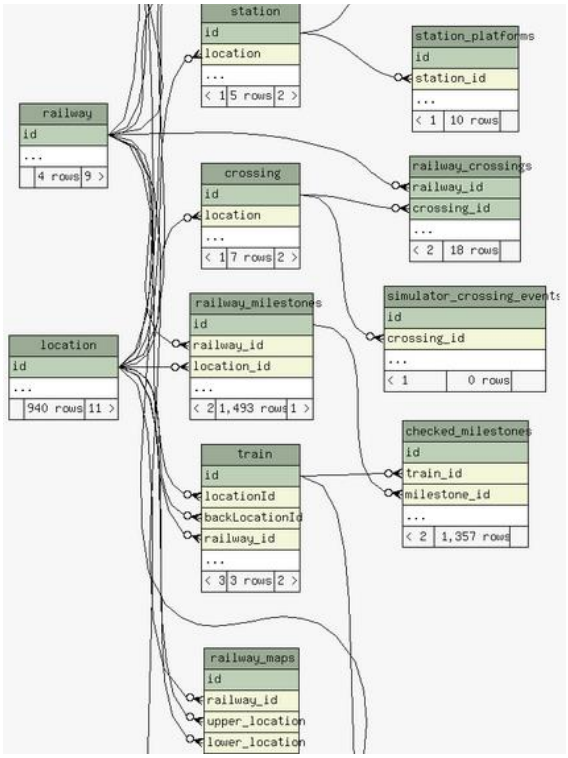

Fig. 1. Part of train simulator data model

The import utility uses the following distance calculation equation to calculate the distance between any two location points A and B using (1).

$$d = r * Math.acos(Math.cos(Math.toRadians(latitudeA)) *Math.cos(Math.toRadians(latitudeB)) *Math.cos(Math.toRadians(longitudeB) - Math.toRadians(longitudeA)) +Math.sin(Math.toRadians(latitudeA)) *Math.sin(Math.toRadians(latitudeB)))) \quad (1)$$

Where "r" is the distance from the center to the surface of the earth, it is a range not an absolute value because of the asymmetry of the earth sphere, but in our system calculations we fixed its value as 6,357 Km which is the effective radius of the earth at sea-level [12].

In this step we need to supply the simulator with vehicle data as its unique identifier, maximum speed, acceleration, and deceleration as basic information to simulate the vehicle movement. In our train simulator example, the train simulator needs additional information such as railway, current station, train direction, etc.

For other simulation cases we will need other type of information for example in a bus simulator we will need information such as bus identifier, traffic average wait time, max speed allowed for some streets, etc.

D. Adding simulator logic

The simulator initial position is set to the start station position, and then it uses the train speed to calculate the run

doi:10.15849/icit.2015.0089

distance. Once the train runs the milestone distance, the simulator uses that milestone location as its current location while communicating its location to the LBS system. The simulator needs to implement a communication protocol with the location application. This includes communicating the simulator location together with the simulator unique identifier.

The logic includes implementing different simulator scenarios to cover all possible business scenarios, in train simulator we built different simulation scenarios such as normal scenario, train broken scenario, split car scenario, etc.

The following train simulation scenarios are built using the simulation model by providing different configurations and custom business logic as in Table II:

TABLE II. SOME GENERATED TRAIN SCENARIOS

Scenario	Business Logic
Normal scenario	No custom logic, the simulator will move the train from start station to end station passing through middle stations and reduce speed at crossings.
Train broken scenario	Keep sending same train location at certain point as train speed equals to zero (broken start location).
Train detached cars scenario	Keep sending same train back location after train passes the split location (split start point).
Train exceed max speed scenario	Exceed train max speed after passing certain point.

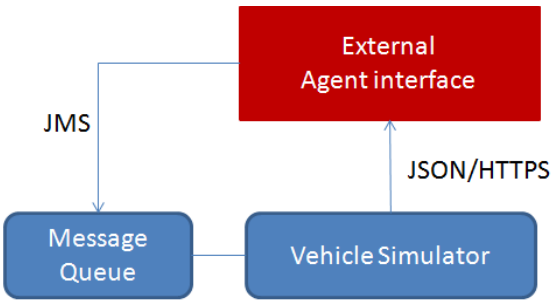

Fig. 2. Simulator inboud/outbound interface

When we execute the simulator we need to feed it up with essential information as initial location, destination location, direction, simulation scenario, speed at each point of time, etc. In our train system the simulator which represents the train will also receives some control agent commands to control the simulator behavior (e.g. speed).

To support this feature we have implemented a messaging queue (using Java Message Service – JMS) where these commands are sent and the simulator keep listening to this messaging queue to execute the received commands, this is important feature to support external control feature of the simulator as shown in fig. 2.

V. VESIMULATOR MODEL

A. *The abstract model*

The VeSimulator model is composed of the following main components; simulator core, data store (database), controller, interfaces (inbound and outbound) and business logic, fig. 3 shows these different components.

The database contains two main data; the location set and the basic simulator information as id, max speed, initial position, status, acceleration and deceleration. The controller part is responsible for lifecycle management operations such as start, stop, pause and resume of the simulator; it exposes these methods for external systems to control the simulator behavior.

The Inbound/outbound connections are the simulator interfaces with the LBS application, the simulator sends the agreed information in JSON object format to the LBS application and receives the agree action JSON object format. The remaining part in the model is the business logic which is the application specific business logic. The simulator core is responsible for retrieving database data, process the simulator logic and any custom business rules, inbound/outbound connections and applies controller commands. This controller component enables the building of a control dashboard to control the simulator.

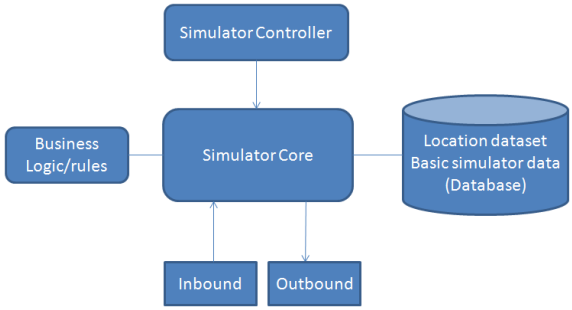

Fig. 3. VeSimulator model

B. *Simulator speed calculation*

As the simulator simulate the movement speed, some points need to be considered here while calculating the final speed at any moment including acceleration and deceleration power, where both are fed to the simulator according to the average values of our simulator model (e.g. average train acceleration and deceleration).

Also supplying the maximum allowed speed on different situations, so for example we have the following speed limitations; train maximum-allowed speed, crossing speed limit, curvature speed limit, switch maximum speed, external speed limit command, etc. and these different speed limitations may overlap pushing the simulator to pick the minimal allowed speed at any point [8].

As the simulator is configured to run each n seconds, the average speed over that period is used to calculate the vehicle run distance (2) and using this distance the existing location point is determined and sent as the current vehicle location. This means the actual point has a rough margin of error that is

doi:10.15849/icit.2015.0089

merely dependent on the frequency of location point sampling rate.

$$run\_distance = direction * abs(V_f + V_0) * n / 2 \quad (2)$$

Where "V_0" is the speed at the beginning of n period, "V_f" is the speed at end of the n period and "direction" is the vehicle direction either forward=1 or backward=-1.

C. Using the simulator

When we deal with location-based services we have to be cautious with the location accuracy. The simulator will keep the accuracy for each location when it communicates it with the application. We should consider the location accuracy and latency in conjugation with current object speed. Having a scenario as we can see in fig. 4; if we have two reported points P1 and P2, both points have an accuracy reported from the GPS device [13]. When we calculate the actual distance between both points we need to consider the worst-case scenario, so for instance if two running vehicles are facing each other the actual distance "d" calculated using (3).

$$d = abs\ (P1 - P2) - (P1_{accuracy} + P2_{accuracy}) \quad (3)$$

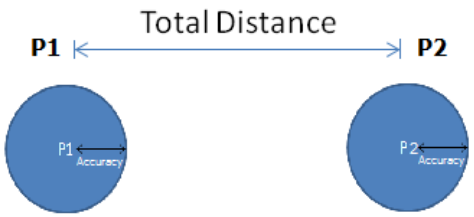

Fig. 4. Total distance between 2 points.

Having IoT location-based applications we need to add one or two additional parameters to our equation to calculate the actual distance; the latency and error buffers. Both values could be configurable and calculated based on the worst-case scenario as well. For example latency buffer which is added to compensate the time spent in device connection to the application, processing time, respond time and execute the action by the device. We can assume this will take 1 second in the average, in that case if the movement speed is 100 Km/Hr the latency is: 100000/3600 = 28 meters. This value represents the run distance by the moving object in that second using its current speed; we need to set this value to a value calculated from the max allowed speed in our business domain to add more safety to the system (e.g. train system).

The error buffer or safety margin is added to the calculations as well to increase the safety of the system or we can combine both in one value. The final equation should looks like (4).

$$d = abs\ (P1 - P2) - (P1_{accuracy} + P2_{accuracy} + latency\_buffer + error\_buffer) \quad (4)$$

VI. Result and Discussion

We have used VeSimulator to create different train simulators and using these simulators in railway control system, we build different test scenarios to test the application ability to control the train in these different running situations. The control system is validated by testing it with different test cases that are constructed using different combination of these scenarios. The system achieved the required target by simulating the train different scenarios and responding back to the railway control system [14].

According to the application nature these scenarios need to be created with different custom logic inside the simulator model so we can cover many real simulation tests to test our application efficiently.

The VeSimulator supports bi-directional interaction so it sends the location position and receive the response back from the tested system; this enables the implementation of different interactive location-based services.

The simulator was tested as an isolated system using the following aspects; simulator controls (start, stop, pause), simulator custom logic scenarios (scenario execution at specific location) and simulator distance against time; all the simulator test cases passed successfully.

Comparing the VeSimulator features with other existing simulators such as TOM can identify the following aspects as shown in table III.

TABLE III. COMPARISON BETWEEN TOM AND VE-SIMULATOR

Aspect	VeSimulator	TOM
Scope	Vehicle location movement simulator	Whole railway system simulator
Speed calculations	Parameters are externally fed	Parameters are calculated according to the given data
External interface	Available	N/A
Outbound data	Yes	Yes
Inbound data	Yes	No
Configurable	Yes	Yes
Support business rules	Yes	No
Application	IoT LBS	Railway simulation

The simulation model is simple and flexible to be adapted as we did in adapting it for train simulation, it is also configurable system so it can be configured in terms of the selected simulation points distance, simulator accuracy is high as being generated from actual collected data, the simulator uses the time based simulation technique to determine the location and finally data re-calibrations can be done without impacting the built system model by re-importing the data again into the model.

VII. Conclusion

The location-based simulator is essential part of developing IoT location-based services; the simulator can improve the

doi:10.15849/icit.2015.0089

speed in developing different location services especially that require business intelligent analysis. It is a generic simulator that can be simply adapted to any LBS applications.

Using the VeSimulator to develop a railway control system achieved the required bi-directional interaction between the simulator and the control system and enabled us to evolve the system to a mature level with less cost and time.

REFERENCES

[1] Ying Sun, "Research on the Railroad Bridge Monitoring Platform Based on the Internet of Things" in *International Journal of Control & Automations*, vol.7, no.1, 2014, pp.401-408.

[2] D. Bandyopadhyay, J. Sen, "Internet of things: Applications and challenges in technology and standardization" in *Wireless Personal Communications*, vol. 58, no. 1, 2011, pp. 49-69.

[3] J. Schiller and A. Voisard, "General aspects of location-based services " in "*Location-Based Services*", Morgan Kaufmann/Elsevier, San Francisco, CA,USA, 2004, ch. 1, pp. 15-32.

[4] F. Reclus and K. Drouard, "Geofencing for fleet and freight management" in *Intelligent Transport Systems Telecommunications*, ITST, 9th International Conference on, Oct 2009, pp. 353-356.

[5] A. Maria, "Introduction to modeling and simulation" in WSC '97: Proceedings of the 29th conference on Winter simulation. Washington, DC, USA: IEEE Computer Society, 1997, pp. 7–13.

[6] GOODMAN C.J., SIU, L.K. and HO, T.K, "A review of simulation models for railway systems*"* in *International Conference On Development in Mass Transit Systems*, 1998, pp. 80-85.

[7] Railway System Center, http://www.railsystemscenter.com/. last retirieved April, 2015.

[8] J. C. Jong and S. Chang, "Algorithms for generating train speed profiles" in *Journal of the Eastern Asia Society for Transportation Studies*, vol.6, 2005, pp.356-371.

[9] Open Rails, http://www.openrails.org/. last retrieved April, 2015.

[10] Manav Singha, Anupam Shukla, "Implementation of location based services in Android using GPS and web services" in *International Journal of Computer Science Issues*, vol. 9, issue 1, no 2, 2012, pp 237-242.

[11] Location Manager APIs– Android Developer, http://developer.android.com/reference/android/location/LocationManager.html. last retrieved April, 2015.

[12] Witchayangkoon, Boonsap, "Elements of GPS precise point positioning", Diss. University of New Brunswick, 2000.

[13] Peter H. Dana, "Global positioning system overview", http://www.colorado.edu/geography/gcraft/notes/gps/gps.html. last retrieved April, 2015.

[14] Mostafa Abdel-Azim, Osama Oransa, "Railway as a thing : new railway control system in Egypt using IoT" in *Science and Information Conference*, London, UK, SAI, 2015, in-press.