Research Article

Integrated Extensible Simulation Platform for Vehicular Sensor Networks in Smart Cities

Xiaolan Tang,1,2 Juhua Pu,1,2 Ke Cao,1,2 Yi Zhang,1,2 and Zhang Xiong1,2

1 State Key Laboratory of Software Development Environment, Beihang University, Beijing 100191, China
2 Research Institute of Beihang University in Shenzhen, Shenzhen 518057, China

Correspondence should be addressed to Xiaolan Tang, tangxl@cse.buaa.edu.cn

Received 31 July 2012; Revised 20 November 2012; Accepted 22 November 2012

Academic Editor: Shan Lin

Copyright © 2012 Xiaolan Tang et al. This is an open access article distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

This paper presents an integrated extensible simulation platform BHU-VSim for vehicular sensor networks (VSNs), which aims to support general simulation environment for typical vehicular applications in smart cities. To deploy urban traffic scenario, we propose a hierarchical object structure to manage entities with different movement models in the network. Furthermore, we design a general data container to present different kinds of data transferred and provide two schemes to generate data packages. Regarding transmission control, the platform includes three components: task scheduling, storage management, and routing deployment. We support importing external routing protocols and configuring relevant parameters to satisfy various transmission requirements. Finally, an instance application, real-time traffic monitoring, and an example of statistical analysis are introduced to prove the practicality and accuracy of the simulator. In one word, as an initiate attempt, our platform provides significant improvement of VSNs’ simulations.

1. Introduction

A vehicular sensor network (VSN) is a kind of network with sensors equipped on fast moving vehicles and provides ubiquitous connectivity among mobile users and efficient vehicle-to-vehicle (V2V) communications. It is widely used in intelligent transportation systems (ITSs) to support various applications, such as safe driving, real-time traffic monitoring, highway toll payment, multimedia resource sharing, and urban mobile surveillance [1]. As vehicles broaden and deepen the extent of data transmission, VSN has been considered as an important improvement in information collection by humans and has attracted increasing research in both academia and industry [2, 3]. Compared with traditional wireless sensor networks, VSN has its own characteristics, such as loose energy constraint, dynamic network topology, unstable connectivity, and geography-based communication patterns.

Nowadays, a brand new theory, interconnecting things together to form a ubiquitous internet of things (IoTs) [4], has been put forward. IoT initiates many new research domains, including smart city. One typical application in smart cities is smart traffic, which aims to avoid traffic jams, save energy resources, and reduce vehicles’ emissions. Although there already exist ITSs in many metropolises to support real-time traffic monitoring, they require many infrastructures, like base stations, lengthy cables, reliable data center, and others. Considering the large overhead to establish such an ITS and its harsh demand about the environment, the concept of IoT motivates us to do research on internet of vehicles (IoVs) and implement traffic services by VSNs with few infrastructures, which is a key application in future smart cities.

In urban VSNs, because of high mobility and small communication radius, connectivity between vehicles cannot have a long duration and end-to-end paths seldom exist. Thus, we consider VSN as a special kind of delay tolerant networks (DTNs) [5] rather than vehicular ad hoc networks (VANETs) [6]. By providing DTN capabilities, some typical challenging situations of vehicular sensor networks may be overcome, such as intermittent connectivity, variable delays, high error rates, and nonexistence of an end-to-end path. That is because DTNs introduce a store-carry-forward paradigm that performs better and uses fewer resources than
those of end-to-end protocols, for each hop of DTNs is optimized individually [7].

As VSN is an emerging field with new variety and fast-moving features, detailed research work is urgent on data representation, data storage, routing protocols, and other aspects. Because experiments on real vehicles are expensive and dangerous, due to high resource consumptions and complicated traffic scenarios, a reliable and functional platform for VSNs’ simulation is starved for. Thus, a simulation platform not only provides simulation scenarios for VSNs’ applications, but also helps to narrow the gap between research work and practical requirements.

However, to the best of our knowledge, little research had been done on a versatile simulation platform for VSNs to integrate DTN specialties. In this paper, we do research on the core problems in VSNs and construct an extensible simulation platform BHU-VSim. It provides a general environment integrated with scenario construction, data management, transmission control, and service optimization. It not only embeds several existing mobility models and routing protocols, but also supports interfaces to import external models and algorithms.

Furthermore, to improve the practicality of the simulator, it supports flexible scenario control, data generation, task scheduling, and package deletion methods. Meanwhile, graphical user interface (GUI) is finely designed to improve user experience. Overall, different applications on VSNs can be easily realized on our platform with relevant configurations. In this paper, we will show how to use our platform.

The rest of this paper is organized as follows. Section 2 presents related work in VSNs and existing simulation platforms. Section 3 details the design objects and system framework of the platform. Sections 4, 5, and 6, respectively, discuss core issues in the platform design, including scenario construction, data management, and transmission control. An instance of real-time traffic monitoring and its application example of statistical analysis are depicted in Section 7. Finally, several improvements discussed for our future work, we conclude the paper in Section 8.

2. Related Work

As a rising hotspot in traffic management, VSN has attracted researchers in academia, industry, and even government. Well-known research institutes and programs include Car2Car-CC by BMW and other famous automobile corporations [8], CarTel by MIT [9], German’s Network on Wheels (NOWs) [10], and Traffic Prediction by IBM. The Federal Communications Commission (FCC) of the US segments a dedicated short range of communication frequency (DSRSC) for intervehicle communications. Research work on VSNs focuses on several aspects, such as network framework, mobility models, routing protocols, storage management, network security, and performance analysis [11, 12]. In this way, VSN can be considered as a mobile network of massive distributed

In VANETs, there already exist several typical simulators, such as GrooveSim [13], TraNS [14], ASH [15], VGSim [16], ITETRIS [17], NCTUns [18], and Veins [19]. Most of them are composed of network simulator, traffic simulator, and even environment simulator. Typical network simulators are ns-3 [20], ns-2 [21], OMNet++ [22], Jist/SWANS [23], V2X, OPNET, and so forth. Typical traffic simulators are SUMO [24], CanuMobiSim [25], VanetMobiSim [26], and so forth. Therefore, we attempt to construct a simulator for VSNs based on DTNs, which also accepts sparse networks through its store-carry-forward paradigm.

Nowadays, most simulators for DTNs are still in progress. One well-known simulator is named opportunistic network environment simulator (ONE), which is constructed by a research group in Helsinki University of Technology [28, 29]. ONE supports embedded and imported routing protocols and mobility models and provides GUI and a set of reporting and analyzing modules. Although ONE is an excellent simulation platform for DTNs, there is still much improvement needed to increase the practicality of the platform, such as data representation and transmission control. In addition, it is not appropriate to simulate VSNs, due to VSNs’ own characteristics, such as vehicular scenario constraints and highly dynamic mobility. Thus, based on ONE, we design our simulation platform to realize the expectations for actual applications in VSNs.

In this paper, we construct an innovative platform BHU-VSim for VSNs’ simulations. It is integrated with several classic algorithms and also supports abundant interfaces for external algorithms, in the domains of mobility model, data representation, routing protocol, and so forth. Meanwhile, we design flexible methods to support system management, such as scenario control, data generation, task scheduling, and package deletion. Thus, our proposal can provide better performance, compared with previous simulators.

3. Platform Architecture

3.1. Design Objectives. From the perspective of applications, main tasks of VSNs are data collection, data management, transmission control, and finally providing timely and reliable services for smart traffic. In this way, VSN can be considered as a mobile network of massive distributed
heterogeneous data. The difficulties to design VSN simulator are complex scenario setting, various data representation, and flexible control mechanisms. In short, we list several objectives in the following for our platform to satisfy.

1. Support complex traffic scenario construction, with well-managed hierarchy of traffic participants and fine scenario regulations.
2. To simulate data collection and transmission in VSNs, the platform includes interfaces to import external algorithms.
3. Users can neatly configure the control schemes according to specific application requirements.
4. Have access to real traffic data from transportation statistics, to analyze and optimize the system design.

3.2. System Framework. In smart cities, as the basic unit, data cell has its life cycle, from data generation to data processing, then to data usage, until data death [30]. From data life cycle standpoint, we design the system framework of our simulator. First of all, we need an actual urban traffic scenario to support environment for data growing. We name this basis as scenario construction module. In data generation process, we use portable and stable sensors to collect information, so we set data collection layer at the bottom of the architecture. Besides, for specific rules about data formats and generation patterns required to guarantee expected data generation, the second lowest layer is data management layer, including data representation and data generation. After data generation, in order to successfully transmit data from one vehicle to another, vehicular transmission control mechanisms strictly influence data processing, including data storage, data transfer, and data update. Thus, this layer is named transmission control layer. When the data reaches its destination, it must be resolved and analyzed according to the requirements of a specific smart service, in order to provide appropriate data for the ultimate use. We name this layer smart service layer. Finally, the target terminal utilizes the data and decides on whether to delete or keep it, which is named terminal application layer. To sum up the previous analysis, VSN system can be segmented into five layers, data collection layer, data management layer, transmission control layer, smart service layer, and terminal application layer, as shown in Figure 1.

In BHU-VSim, we design a hierarchical object structure to construct scenarios with fast-moving vehicles, slow-moving pedestrians, static roadside infrastructure, and other entities. In data collection layer, sensors equipped on vehicles, roadside infrastructures, and mobile devices with pedestrians gather surrounding information. In data management layer, various data representation and flexible data generation are discussed. Transmission control layer contains task scheduling, storage management, and routing deployment components. Smart service layer supports basic VSNs’ services, such as safe driving and traffic management. The top layer is terminal application layer, which satisfies different requirements of different terminals. Besides scenario setting and performance evaluation, our research mainly covers the data management, transmission control and smart service layers. In our platform, we focus on four modules as listed in the following.

1. Scenario construction: to construct vehicular network scenarios, we introduce the digital map pattern to load instance map, and manage entities in a hierarchical structure to support flexible entity loading. Each entity has its characteristics and can adjust its movement states.
2. Data management: this module consists of two parts, data representation and data generation. Firstly, we design a general data container to present different kinds of packages transferred in the network. Besides, the platform allows periodical or random generation of data packages.
3. Transmission control: BHU-VSim provides task scheduling methods and package dropping methods to improve transmission and storage efficiency. Meanwhile, it is integrated with several typical routing protocols and has interfaces to import external protocols.
4. Service optimization: analyzing basic information of the network and data received from traffic entities, vehicles compute relevant parameters to indicate service quality and adopt corresponding measures to optimize the quality.

To be noted, service optimization is used to prepare rough data well for terminal applications. Via computing service parameters which serve for the applications and defining optimization methods to optimize the simulation results of service parameters, this module ultimately realizes the value of data and provides high-quality service. Here, we see service optimization module always has strong association with specific applications. For an instance, a multimedia resource sharing application may require selection and integration of different data blocks, while a traffic monitoring application maybe needs the latest vehicular flow rate at each waypoint. Different requirements lead to a huge difference between the implementations of service optimization modules in these two applications. Therefore, to introduce general models in our simulator, we omit further discussion about service optimization in the next sections. However, an instance of service optimization module will be depicted in Section 7. In Sections 4–6, we describe core issues in the implementation of scenario construction, data management, and transmission control modules in detail.

4. Scenario Construction

As traffic status in urban areas is always complicated, exactly simulating an urban traffic scenario is often difficult. Even previous work on VANET simulators could not provide adequate models and convenient extensions to create various transportation participants. Additionally, as ONE is not specially designed for vehicular networks, it misses several
4.1. Map Format. To simulate transportation in smart cities, we focus on the road information in the urban map rather than buildings, farmland, and so forth. Thus, each map is composed of many paths, and each path has a list of road waypoints. Similar with ONE, we select .wkt digital map file to be the standard map formats. To better simulate roads in cities, we add multilane and directions for .wkt file. Specifically, we allow users to set the numbers of lanes in two directions in the map file. Default numbers of lanes in two directions are both 1. The direction in line with the order of road waypoints is named positive direction, remarked “+.” The other direction is called negative direction, remarked “−.” The serial number of each lane is started from the roadside (1) to the middle of the path (n). Each lane has its lane number, such as “+1” and “−2.” To be noted, real city maps in .wkt format can be imported into the system directly.

4.2. Entity Management. To support unified and extensible traffic entity management, we propose a hierarchical entity structure. Entities with similar characteristics comprise one group with the common attributes, and similar groups are also clustered together, in order for convenient management and deployment. The system automatically constructs entity groups in batch and then regulates entities with special features different from its group. Each entity has three kinds of attributes, including basic attributes (ID, name, count, etc.), mobility attributes (mobilityModel, initialLocation, path, speed, waitTime, movingTime, etc.), and communication attributes (commRange, dataRate, bufferSize, router, packages, etc.). Users can customize these attributes for a particular application in the class file Entity.

BHU-VSim is embedded with several movement models for traffic participants, such as CarMovementModel and BusMovementModel. To improve flexibility of the platform, external movement models are supported by extending MobilityModel class and overriding at least two methods setInitialLocation() and setPath(). Additionally, actual mobility files can be used in the system by calling methods with parameter mobilityFile.

Besides, in BHU-VSim, we implement IDM car-following model [31] and MOBIL lane-changing model [32] to simulate vehicular mobility constraints. With configuration sentences carFollowingModel = IDM and laneChangingModel = MOBIL in the mobility attributes, users can use these models.

4.3. Traffic Status Adjustment. To improve realness of BHU-VSim, we use traffic status adjustments to set jams in the scenario. We propose the configuration of influence sphere to improve the simulation accuracy, compared with obstacles defined in ASH [15]. For instance, a traffic control measure may just stop private cars passing at one direction. Relevant methods to set a block are shown in Table 1. Among
the methods, setPaths() and setEntities() are required. The default lanes are all lanes, and the default speed is 0.

The scenario construction module of BHU-VSim supports urban map with multilane paths, has a hierarchical structure to manage transportation participants, and also allows setting road blocks. Although these specialties help to construct more real traffic scenario, there are still many problems to be solved, such as the traffic light model, the overlapping issue due to overpass, and the building’s interference with V2V communication. More actual 3D model may be helpful to overcome these problems in the future.

5. Data Management

Although previous network simulators seldom discuss the data management program, it is very significant to define data when simulating an actual application in VSNs, for data being the key element in the network communication. In BHU-VSim, we provide a general data container to represent different kinds of data packages for different applications. Moreover, we design two methods to support periodical generation and random generation of data packages.

### 5.1. Data Representation

To improve the flexibility and availability of BHU-VSim, we design a general data container to present and differentiate various data. We partition a package into header section and data section. For a specific application, the data package can be defined by inheriting class DataPackage, setting necessary attributes, and overriding the method Entity.createPackage() to assign values for the attributes in the package.

In our simulator, we have designed three kinds of data packages for typical services in VSNs, entity status data, digital data, and multimedia data. Entity status data presents this entity’s basic information, moving state, and position relations with others, in order to monitor traffic and support safe driving. Digital data is always taken from different kinds of sensors to record temperature, humidity, and so forth. These sensors use digital to indicate the traffic states and road states. Multimedia data is designed for entertainment applications, including file format, minimum data rate, recovery rate, and content.

Besides data messages, control messages are required in almost every application. They always have particular requirements about transmission and analysis. Thus, we leave the discussions about control messages for our future work.

### 5.2. Package Generation

In BHU-VSim, we provide two interfaces to periodically or randomly generate data packages. By calling the method setProdPkgGntTime(float start, float end, float interval), this entity will generate one data package every interval rounds from the round start to end. By calling methods setRandPkgGntTime(float start, float end, int count), this entity will randomly generate count data packages from start to end. Although these methods are not complete, they are sufficient for usual data generation.

In BHU-VSim, we define the format of data package and present three typical instances. Besides, two data generation methods enrich the simulation scope. In this module, there are much more research needed on the control package formats and simplified data structure design, in order to improve the accuracy and efficiency of the simulation.

6. Transmission Control

How to successfully transfer a data package from its source to its destination is a key problem in network simulation. In VSNs, considering the unstable connection, when two entities meet with each other (they are in each other’s communication range), they should grasp this valuable opportunity to transfer data between them. In previous work, although some simulators are embedded with classical routing protocols and storage management policy, they are incomplete and difficult to extend. After analyzing the influence factors in data transmission, we think that three basic problems should be solved: (1) which entity transfers first, and which package is transferred early; (2) what routing protocol to use for the package transmission; (3) if no enough storage left to receive the data, which package is dropped early to save space. To handle these issues in an extensible way, we introduce the task scheduling, storage management, and routing deployment policies.

### 6.1. Task Scheduling

To provide equal chance for each entity to transfer data, we propose random entity selection policy to select one meeting entity to send data first. Then, we use scheduling model to compute the transmission priority of each package in the sender entity, to improve the transmission efficiency.

Using the random entity selection policy, each meeting entity generates a random value in the interval (0, 1). Then, the entity with the largest value transfers first, and the entity having the second largest value transfers next, until each of them transfers one package. If there is still a transfer chance, the second round of transmission with the same sequence takes place. In this way, each entity has an equal chance to send data. Here we assume that all the nodes obey this policy, without mendacity. How to realize this assumption is left for research on network security.

When one entity has a chance to transmit data, firstly it computes scheduling priorities of all the packages in its storage and then selects the package having the highest priority to transfer. Here, we use basic scheduling attributes to stand for those attributes defined in the ScheduleModel
class. The method `computePriority()` in this class uses the values of these attributes to compute the scheduling priority.

The customized scheduling model should inherit from the class `ScheduleModel`, setting basic scheduling parameters and overriding the `computePriority()` method.

Traffic entities use the method `setScheduleModel()` to set scheduling model for their packages and assign values to basic scheduling attributes in the model. Here, we suggest the configuration to be set in batch if one or more groups of entities have the same policy.

6.2. Storage Management. Taking VSN as a distributed storage system with limited storage space in each vehicle, when the buffer is overfilled, the vehicle drops data according to packet dropping rule, in order to make space for new coming data.

To increase the flexibility of BHU-VSim, we design a package dropping model based on selected attributes. The implementation is similar to the task scheduling policy. Basic dropping attributes are defined in the `DropModel` class, and the method `computePriority()` uses the values of these attributes to compute the dropping priority. Packages with higher priorities are dropped earlier. Traffic entities use the method `setDropModel()` to set drop model for their packages.

6.3. Routing Deployment. In urban traffic, each entity has its routing protocol to transfer data packages. For instance, the fast-moving vehicle may use Prophet protocol to transfer data to a specific destination, while road-side infrastructure may flood data to passing buses. Furthermore, to satisfy complicated requirements, we support flexible deployment according to traffic states. For example, in a crowded area, vehicles use Spray and Wait routing protocol to keep the number of package replica within a limit, while in some sparse area, Epidemic routing is selected to increase the probability of successful transmission. Thus, we support routing deployment based on different entity groups and traffic states.

Here, traffic state attributes are those attributes of the entity, which affect it to select routing protocol. Each entity sets its routing protocol with the method `setRouter()` by considering the values of these traffic state attributes.

BHU-VSim has been integrated with several typical routing protocols, including EpidemicRouter, SprayWaitRouter, and ProphetRouter classes. The interface to import new routing protocols is also provided. We use basic routing attributes to stand for those attributes defined in a new routing class, and the method `update()` in the class uses the values of these attributes to make routing decision, such as whether to transmit this message and how many replica to transmit (the usage of the number of replica can be obtained from SprayWaitRouter class). Users can create a new `Router` class inheriting the `BasicRouter` class, define the basic routing attributes, and override the method `update()`, to implement customized routing decision making process.

Besides the control methods described previously, we have concrete operation process in the core of the platform. In other words, user-defined control mechanisms are well integrated into BHU-VSim, if they are configured using the interface methods as defined previously. Although our improvement has great significance to simulate actual applications in VSNs, there are still more problems about transmission control to be handled, such as data inconsistency, data corruption, and data acknowledgment. Besides, to simplify and clarify the structure of the simulator, we omit the handshake process when establishing connection. In our future work, we should gradually improve the fidelity of the transmission process.

7. Performance Evaluation

7.1. An Instance: Real-Time Traffic Monitoring. As smart traffic is attracting more and more attentions from the government to individuals, we introduce one instance application, real-time traffic monitoring in VSNs, which aims to optimize vehicles’ path selection to avoid traffic jams through V2V communication, without any roadside infrastructures.

In scenario construction module, to set urban traffic scenario for this application, we deploy two vehicle groups, VGroup1 and AGroup1, in the 500 × 500 map zone during the simulation rounds [0, 500]. VGroup1 has 100 cars moving with CarMovementModel movement model, and AGroup1 has 12 buses with BusMovementModel mobility model. Here, buses in AGroup1 gather traffic congestion information, because of their stable movement paths, fixed departure interval, and high network coverage. Cars in VGroup1 just communicate with AGroup1 to get the traffic state. In this way, the traffic state can be disseminated in the scenario with low transmission overhead. Here, we use our traffic status adjustment method to set one jam on the shortest path from the start point to the destination point. Configuration interfaces are listed in Table 2.

In data management module, we use the vehicle driving speed to indicate traffic congestion degree. Specifically, if the vehicle passing by one path runs fast, the traffic there is smooth and clear; if the vehicle can hardly move forward, there is possibly a traffic jam. In the implementation, we use entity status data to represent the data package. To save storage space, the entity status data just contains a quaternion, including recorder entity’s ID, location, velocity, and record time. Concerning data generation pattern, each bus generates 1 package in every 5 rounds from the 0th round to the 400th round. Relevant methods are listed in Table 2.

In transmission control module, to improve transmission efficiency, we utilize task scheduling and package dropping policies. In the task scheduling policy, we use two attributes in data package p, record time and location, to be basic scheduling attributes. Packages with higher record time and more adjacent location are transferred earlier than others. In the package dropping policy, we use record time to be basic dropping attribute. Packages with lower record time are dropped earlier than others, to save space for new data. For the routing protocols, we use the embedded protocols EpidemicRouter and ProphetRouter. The velocity of entity is traffic state attribute. When the velocity is equal to or larger than 20, which means the traffic state is excellent, buses use
Table 2: System configurations for the instance application.

| Module                     | Function                              | Method                                                                 | Remark                                             |
|----------------------------|---------------------------------------|------------------------------------------------------------------------|----------------------------------------------------|
| Scenario construction      | Set mobility models for entities       | VGroup1.setMobilityModel(Car-MovementModel)                            |                                                    |
|                            |                                       | AGroup1.setMobilityModel(Bus-MovementModel)                           |                                                    |
|                            | Set a traffic block $b$                | $b$.setPaths(path1, path2)                                             |                                                    |
|                            |                                       | $b$.setEntities(VGroup1, AGroup1)                                      |                                                    |
|                            |                                       | $b$.setSpeedRange(0, 5) scenario.addBlock($b$)                         |                                                    |
| Data management            | Represent data package $p$             | AGroup1.createPackage()                                                | Data: ID, location, velocity, and record time      |
|                            | Set generation time of data package $p$ | AGroup1.setProdPkgGntTime (0, 400, 5)                                  |                                                    |
| Transmission control       | Set task scheduling policy $ts$        | VGroup1.createScheduleModel($ts$)                                      | Basic scheduling attributes: record time and location |
|                            |                                       | AGroup1.createScheduleModel($ts$)                                      |                                                    |
|                            |                                       | $ts$.computePriority()                                                 |                                                    |
|                            | Set package dropping policy $pd$       | VGroup1.createScheduleModel($pd$)                                      | Basic dropping attributes: record time             |
|                            |                                       | AGroup1.createScheduleModel($pd$)                                      |                                                    |
|                            |                                       | $pd$.computePriority()                                                 |                                                    |
|                            | Set routing protocols for buses in AGroup1 | AGroup1.setRouter()                                                  | Traffic state attributes: velocity                |

**EpidemicRouter** to propagate information; otherwise, when the traffic is jammed, they use **ProphetRouter** to decrease transmission overhead. Table 2 shows the configuration methods.

To avoid transmitting traffic states of irrelevant paths, buses just collect data about the 5 recommended shortest paths. Cars obtain the traffic information when they communicate with buses. Then, cars can optimize their path selection by analyzing the traffic states.

In service optimization module, with geographical data and traffic data, we put forward the path selection method. To avoid traffic jams, cars exclude the initial recommended paths including jams first and then select the one with the shortest path as the final path. However, maybe all the initial paths have jams. In this case, another 5 short paths are computed, and another round of jam rejection is implemented, until one path excluding jams is found. The GUI of BHU-VSim is designed to show the scenario and transmission state, as presented in Figure 2(a). The result of path selection to avoid jams is shown in Figure 2(b).

For other applications, users just need to configure the scenario, define the data, and override relevant control methods, in order to provide required services. Overall, we can conclude that the simulation platform BHU-VSim is easily configurable and extensible. It can satisfy different requirements of various applications in VSNs.

7.2. Statistical Analysis. In our simulator, we have data analysis module to compare the statistic results of different algorithms. Independent of the traffic monitoring instance, we test the performance of typical routing protocols under limited storage in VSNs, including **Epidemic**, **Prophet**, and **Spray and Wait** routing protocols. Configurations are listed in Table 3. We set the shortest path map-based movement model and FIFO package dropping model for vehicles.

Table 3: Configurations for the example of statistical analysis.

| Parameter                  | Value                      |
|----------------------------|----------------------------|
| Network area               | $5000 \times 5000$ m$^2$  |
| Number of vehicles         | 100                        |
| Simulation time            | 12 h (1 round stands for 1 second) |
| Communication radius       | 100 m                      |
| Mobility model             | The shortest path map-based model |
| Mobility speed             | Random in (10, 20) m/s     |
| Data rate                  | 2 Mbps                     |
| Size of data message       | 256 KB                     |
| Size of storage space      | 5 MB                       |
| Data generation            | 1 message every 15 min by each vehicle |
| Package dropping model     | Fist in first out (FIFO)   |

The evaluating indicators are the delivery ratio (ratio of the number of successful delivery messages to the number of total messages), average delay (average delay of successful delivery messages), average hop (average hop of successful delivery messages), and average number of replicas (ratio of the number of total replicas to the number of messages). The simulation results are shown in Figure 3.

From Figure 3, we see that in general, **Spray and Wait** routing protocol performs best among the three protocols. It has the highest delivery ratio, the lowest average delay, the lowest average hop, and the lowest average number of messages. The reasons lie in two parts. (1) As we use the shortest path map-based movement model, the advantages of **Prophet** cannot be shown out. This is because the mobility model causes irregular counter probabilities, which seriously interfere the forwarder selection in **Prophet** routing protocol. (2) The **Epidemic** routing protocol uses flooding method to disseminate data, so it leads to heavy resource consumption.
of storage spaces. In this test, we set limited storage space at vehicles and use FIFO dropping rule to manage storage space. Thus, Epidemic leads to frequent change of the packages before they are forwarded. Thus, the performance of Epidemic routing protocol deteriorates.

From qualitative analysis, the simulation results are in line with theoretical derivation. Thus, it testifies the correctness of statistic results using our platform to some extent and further shows the practicality of our platform in algorithm performance comparison.
In this section, we introduce two typical instances. One is a real application, real-time traffic monitoring, in order to show the usage of our simulator and indicate the practicality and convenience of our platform. The other is statistical analysis of three routing protocols, in order to testify the accuracy of simulation results qualitatively. However, more actual experiments are required in our future work to further evaluate the performance of BHU-VSim quantitatively.

8. Conclusion

In this paper, we construct an integrated extensible platform named BHU-VSim, to support simulation environment for typical applications in VSNs. To deploy urban traffic scenario, we define standard map files with multiline paths, propose a hierarchical object structure to manage traffic entities, and provide traffic status adjustment to set jams in the scenario. Furthermore, to manage data, we define data package formats and support periodical and random data generation methods. Considering the key problems in transmission process, BHU-VSim supports flexible task scheduling policy, package dropping policy, and routing deployment method. Besides the embedded mobility models and routing protocols, it has abundant interfaces to import external models and protocols. Finally, an instance of real-time urban traffic monitoring application and an example of statistical analysis are introduced to illustrate the usage, practicality and accuracy of the simulator. In one word, BHU-VSim is a valuable attempt to construct an integrated and extensible simulation platform for VSNs.

However, there is still much profound research to be done to improve our simulator. As the channel model is crucial to simulate V2V communication, we will continue our research to integrate the models in physical layer and MAC layer into BHU-VSim. Additionally, more experiments in actual vehicular sensor networks are needed to test the simulator’s performance and adjust it well for real-world scenarios.

Acknowledgments

The authors gratefully acknowledge the support from the Natural Science Foundation of China (61272350 and 61173009), National High Technology Research and Development Program of China (2011AA010502), International S&T Cooperation Program of China (2010DFB13350), Doctoral Fund of Ministry of Education of China (20091102110017), Science Foundation of Shenzhen City in China (JCYJ20120618170520900), and Fundamental Research Funds for the central universities.

References

[1] Y. Zhu and Y. Jian, “A game-theoretic approach to anti-jamming in sensor networks,” in Proceedings of the 16th IEEE International Conference on Parallel and Distributed Systems (ICPADS ’10), pp. 617–624, December 2010.
[2] M. Gerla and L. Kleinrock, “Vehicular networks and the future of the mobile internet,” Computer Networks, vol. 55, no. 2, pp. 457–469, 2011.
[3] X. Li, Research on the key techniques of vehicular sensor networks and applications [Dissertation], Shanghai Jiao Tong University, 2009.
[4] L. Chen, M. Tseng, and X. Lian, “Development of foundation models for Internet of Things,” Frontiers of Computer Science in China, vol. 4, no. 3, pp. 376–385, 2010.
[5] M. J. Khbazz, C. M. Assi, and W. F. Fawaz, “Disruption-tolerant networking: a comprehensive survey on recent developments and persisting challenges,” IEEE Communications Surveys & Tutorials, vol. 14, no. 2, pp. 607–640, 2011.
[6] H. Hartenstein and K. P. Laberteaux, VANET: Vehicular Applications and Inter-Networking Technologies, John Wiley & Sons, 2010.
[7] P. R. Pereira, A. Casaca, J. J. P. C. Rodrigues, V. N. G. J. Soares, J. Triay, and C. Cervello-Pastor, “From delay-tolerant networks to vehicular delay-tolerant networks,” IEEE Communications Surveys & Tutorials, vol. 14, no. 4, pp. 1166–1182, 2012.
[8] L. Franck and F. Gil-Casteñeda, “Using delay tolerant networks for Car2Car communications,” in Proceedings of the IEEE International Symposium on Industrial Electronics (ISIE ’07), pp. 2573–2578, June 2007.
[9] The CarTel project, http://cartel.csail.mit.edu/.
[10] NOW: Network on Wheels, http://www.network-on-wheels.de.
[11] “UDel Models for Simulation of Urban Mobile Wireless Networks,” http://udelmodels.eecis.udel.edu/.
[12] J. Harri, F. Filali, and C. Bonnet, “Mobility models for vehicular ad hoc networks: a survey and taxonomy,” IEEE Communications Surveys & Tutorials, vol. 11, no. 4, pp. 19–41, 2009.
[13] R. Mangharam, D. S. Weller, D. D. Stancil, R. Rajkumar, and J. S. Parikh, “GrooveSim: a topography-accurate simulator for geographic routing in vehicular networks,” in Proceedings of the 2nd ACM International Workshop on Vehicular Ad Hoc Networks (VANET ’05), pp. 59–68, September 2005.
[14] M. Pioriowski, M. Raya, A. Lugo, P. Papadimitratos, M. Grossglauser, and J. P. Hubaux, “TraNS: realistic joint traffic and network simulator for VANETs,” in Proceedings of the ACM SIGMOBILE Mobile Computing and Communications Review (MC2R ’07), 2007.
[15] K. Ibrahim and M. C. Weigle, “ASH: application-aware SWANS with highway mobility,” in Proceedings of the IEEE Conference on Computer Communications (INFOCOM ’08), pp. 1–6, April 2008.
[16] B. Liu, B. Khorashadi, H. Du, D. Ghosal, C. N. Chuah, and M. Zhang, “VGSim: an integrated networking and microscopic vehicular mobility simulation platform,” IEEE Communications Magazine, vol. 47, no. 5, pp. 134–141, 2009.
[17] V. Kumar, L. Lin, D. Krajewicz et al., “ITETRIS: adaptation of ITS technologies for large scale integrated simulation,” in Proceedings of the 71st IEEE Vehicular Technology Conference (VTC ’10), May 2010.
[18] S. Wang and C. Lin, “NCTUns 6. 0: a simulator for advanced wireless vehicular network research,” in Proceedings of the IEEE Vehicular Technology Conference (VTC ’10), 2010.
[19] C. Sommer, R. German, and F. Dressler, “Bidirectionally coupled network and road traffic simulation for improved IVC analysis,” IEEE Transactions on Mobile Computing, vol. 10, no. 1, pp. 3–15, 2011.
[20] Network Simulator ns-3, http://www.nsnam.org/.
[21] Network Simulator ns-2, http://isi.edu/nsnam/ns/.
[22] OMNet++, http://www.omnetpp.org/.
[23] R. Barr, Z. J. Haas, and R. van Renesse, “JiST: embedding simulation time into a virtual machine,” in Proceedings of the EuroSim Congress on Modeling and Simulation, 2004.

[24] SUMO, http://sumo.sourceforge.net.

[25] CanuMobiSim, http://canu.informatik.uni-stuttgart.de.

[26] J. Härri, F. Filali, C. Bonnet, and M. Fiore, “VanetMobiSim: generating realistic mobility patterns for VANETs,” in Proceedings of the 3rd ACM International Workshop on Vehicular Ad Hoc Networks (VANET ’06), pp. 96–97, Los Angeles, Calif, USA, September 2006.

[27] M. Zhang and R. S. Wolff, “A border node based routing protocol for partially connected vehicular ad hoc networks,” Journal of Communications, vol. 5, no. 2, pp. 130–143, 2010.

[28] A. Keranen and J. Ott, “Increasing reality for DTN protocol simulations,” Tech. Rep., Networking Laboratory, Helsinki University of Technology, July 2007.

[29] A. Keranen, J. Ott, and T. Karkkainen, “The ONE simulator for DTN protocol evaluation,” in Proceedings of the 2nd International Conference on Simulation Tools and Techniques (SIMUTOOL ’09), March 2009.

[30] Z. Xiong, W. Luo, L. Chen, and L. M. Ni, “Data vitalization: a new paradigm for large-scale dataset analysis,” in Proceedings of the 16th IEEE International Conference on Parallel and Distributed Systems (ICPADS ’10), pp. 251–258, December 2010.

[31] M. Treiber, A. Hennecke, and D. Helbing, “Congested traffic states in empirical observations and microscopic simulations,” Physical Review E, vol. 62, no. 2, pp. 1805–1824, 2000.

[32] A. Kesting, M. Treiber, and D. Helbing, “MOBIL: general lane changing model for car-following models,” in Proceedings of the Transportation Research Board Annual Meeting, January 2007.