CHALMERS

Forecasting Short Term Traffic Conditions using Vehicular Ad-hoc Network

Master of Science Thesis in Networks and Distributed Systems

YANG YANG, XIAOCHU WANG

Department of Computer Science and Engineering
CHALMERS UNIVERSITY OF TECHNOLOGY
UNIVERSITY OF GOTHENBURG
Göteborg, Sweden, November 2009

Forecasting Short Term Traffic Conditions using Vehicular Ad-hoc Network

Yang Yang, Xiaochu Wang

Examiner: Marina Papatriantafilou

Department of Computer Science and Engineering
Chalmers University of Technology
SE-412 96 Göteborg
Sweden
Telephone + 46 (0)31-772 1000

Department of Computer Science and Engineering
Göteborg, Sweden November 2009

Abstract

Vehicular ad hoc networks (VANETs) represent a particularly challenging class of mobile ad-hoc networks (MANETs) which enable vehicles to communicate with each other. In VANETs, vehicles are restricted in their range of communication and tend to move in an organized pattern. A virtual stationary automaton (VSA) is a static infrastructure consists of fixed, timed virtual machines, and they are tiled over the entire plane. Our project can be categorized as an information dissemination system which applies in VANETs by providing surrounding information to road user. We use the pre-defined VSAs to predict traffic condition in each VSA region in a short given term. The challenges are how to get the information to predict and how to analyze the data. We implemented new message in VSA deployment and studied the performances of our design in a mathematic way. The application domains of our project include dynamic geographical information system (GIS) and improved communication networks.

Acknowledgements

We are grateful to the project for giving us the opportunity of research presented in this thesis, and would like to extend a thank you to all the project participants for their valuable contributions. In particular, we want to express our appreciation to Elad Schiller for providing us this project and all his assistance throughout our thesis work.

With our deepest gratitude, we thank Marina Papatriantafilou and Andreas Larsson, for taking us under their wings and providing us with endless support through the many ups and downs over the last few months.

We would also like to thank Shan Liu, Ilhan Uludag, Ning He Gongxi Zhu, and the colleagues at the Group of Distributed Computing and Systems in Chalmers University of Technology, for creating a positive and friendly working environment for us.

Table of Contents

1. Introduction

1.1. Project Motivation

Vehicular networks, nowadays, typically support three categories of application: information dissemination, driver assistance, and coordinated driving. A typical information dissemination system is a kind of local information system which is a significant application domain for vehicular networks. Information dissemination system provides information to road users on their surroundings.

Our project aims to develop such an information dissemination system to provide road information to drivers. The information we provided might be adapted to the context of the users. For example, predicting number of cars in a fixed region, then the driver assistance system can make a better routing strategy, furthermore, bad traffic condition can possibly be avoided thus drivers can save both time and money.

1.2. Terminology

- Mobile Ad Hoc Network (MANET): It is an autonomous collection of mobile users communicating over a relatively bandwidth-constrained wireless link in high dynamic environmental [1].
- Vehicular Ad Hoc Network (VANET): It is a form of Mobile ad-hoc network, providing vehicle-to-vehicle communications; as well as vehicle-to-infrastructure communications.
- Java in Simulation Time (Jist): It is a high-performance discrete event simulation engine.
- Scalable Wireless Ad hoc Network Simulator (Swans): It is a scalable wireless network simulator built atop the Jist platform.
- Street random waypoint (STRAW): It aims to implement and deploy network protocols and services for vehicular ad-hoc networks (VANETs) on simulation for evaluation. STRAW provides more accurate simulation results by using a vehicular mobility model on real street map, based on real vehicular traffic conditions [6].
- Virtual Stationary Automata (VSA): The static infrastructure consists of fixed, timed virtual machines, called Virtual Stationary Automata (VSAs) that are tiled over the entire plane. Each VSA represents a predetermined geographic area and has broadcast capabilities similar to those of the mobile nodes, allowing nearby VSAs and mobile nodes to communicate with one another [3].
- Correlation: Correlation indicates the strength and direction of a linear

relationship between two random variables.

- Map: A map (also associative container, mapping, hash, dictionary, finite map, and in query-processing an index or index file) is an abstract data type composed of a collection of unique keys and a collection of values, where each key is associated with one value (or set of values). The relationship between a key and its value (or set of values) is called mapping or binding.

1.3. Background

1.3.1. MANET

A MANET (Mobile Ad-hoc Network) is an autonomous collection of mobile users communicating over a relatively bandwidth-constrained wireless link in high dynamic environmental [1]. MANETs can be considered as a kind of wireless ad hoc networks which is a routable networking environment. MANETs is a type of mesh network, whereas many other mesh networks are not mobile and wireless. Mobile nodes in MANET have to set up themselves in a possible short-lived network for communications when they are needed.

1.3.2. VANET

VANET (Vehicular Ad Hoc Network) is a form of Mobile ad-hoc network, providing vehicle-to-vehicle communications, and also vehicle-to-infrastructure communications. The main goal of VANET provides service for the vehicles within the Ad-Hoc networks. A special electronic device is placed inside each vehicle which provides Ad-Hoc Network connections. Each vehicle equipped with VANET device is a node in the Ad-Hoc network, which can receive and relay others messages through the wireless network. This network tends to operate without any infra-structure or legacy client and server communication. Most of the concerns of interest to MANETs are of interest in VANETs, but there still some details are different. First, MANET nodes are moving at random, vehicles tend to move in an organized fashion. Second, most vehicles are restricted in their range of communication. VANET integrates on multiple ad-hoc networking technologies such as WiFi IEEE 802.11 b/g, WiMAX IEEE 802.16, Bluetooth, IRA, ZigBee for communication between vehicles on dynamic mobility.

The main applications for vehicular networks as follow:

1. Information dissemination: providing services for all passengers of a vehicle
2. Driver assistance, offering support for driving maneuvers
3. Coordinating driving, controlling the trajectory of vehicles

Local information system, is a significant application domain for vehicular networks, is the provision of information to road users on their surroundings [2]. Some of these systems are targeted to specific usages.

1.3.3. VSA

In distributed systems, people normally aggregate nodes to construct distributed structures. By increasing the degree of information locality, it may improve system scalability. There usually exists a predefined clustering to make the wireless communication networks simple and efficient. The VSA has been implemented in VANET as a variant of these techniques. It is a programming abstraction that considers the construction of a distributed structure of VSAs. The VSAs are located at pre-specified regions and tile the plane. The regions are organized according to predefined distributed structures that assist in improving the locality of operations and simplify computation of global functions [4]. So, VSA is used to guarantee the reliability of regional services had been provided.

Considering the mobile nodes can drift apart at any time, a straightforward clustering of the mobile nodes cannot provide the locality and scalability properties. But self-stabilization is able to recover from an arbitrarily corrupt state. These properties satisfy the system which needs long-live and stable. The VSA abstraction simplifies the above complications and allows to pre-specified "clusters" of virtual stationary nodes. Each region that is associated with a particular VSA is a distributed structure. The participator depends on the existence of mobile nodes within the region and the frequency of failure. The result is a "best effort" construction that is maintained under probabilistic assumptions.

1.3.4. Simulation

Jist (Java in simulation time) is a high-performance discrete event simulation engine that runs over a standard Java virtual machine, and all components are pure Java. The kernel of Jist is strict partitioning of a simulation into entities. Method invocations on objects marked as entities represent simulation events. No explicit event queue, but virtual, explicit time progress. It is a prototype of a new general-purposed approach to build discrete event simulators, called virtual machine-based simulation which unifies the traditional systems and language-based simulator designs. The basic idea of Jist is to convert virtual machine into simulation platform, and it introduces virtual time. Running Jist invokes Rewriter which modifies Java bytecode to introduce simulation time semantics. When Jist invokes simulation program, rewritten program interacts with simulation kernel, the virtual time progress is independent of program progress (instructions take zero virtual time). Time is advanced explicitly via JistAPI.sleep(), and time synchronization between Entities on method invocation [5].

1.4. Related work

There are so many different methods for traffic prediction, some technology collects a large of historical traffic flow information data and analysis them to acquire useful traffic pattern, as suggested in [7]. This approach in VANET may results the large amounts of data arrive in receiver simultaneously. In the thesis [8] present self-stabilizing middleware service in ad-hoc networks: group membership, group multicast, and resource allocation. The thesis [9] has implement the approach present in the thesis [8], appears the pre-specified, fixed-location and self-stabilizing virtual stationary automata (VSA). The thesis [4] implements the message passing in the VSA, which make the traffic prediction in a given area can be easily implement. Using the pre-defined VSA to approach the traffic prediction will decrease the possibility of communication messages arriving at receivers simultaneously, and also decrease the requesting for the receiver's processing quantity, and also reduce some complicated statement may happen because of the self-stabilizing.

1.5. Specific aims

In this project, we mainly focused on the short-term traffic prediction in simulation. There is a prediction algorithm, which is used to predict traffic condition, will be developed and implemented in simulation.

In order to do a traffic forecasting, we need to count the number of vehicles in a fixed geographic area at certain time points. The number of vehicle coming from/to the fixed geographic area during a certain time period is also needed for forecasting. We can re-use the VSA implementation to set our fixed and timed virtual machine.

The Street Random Way (STRAW) mobility model provides more accurate simulation results by using a vehicular mobility model on real street map than other mobility models [6]. In our project, we evaluated our algorithm by using such a mobility model to get our result and conclusion.

The main challenges of our project are below:

1. Information: The VSA infrastructure does not implement any functions to get the information of vehicle's number, and also the number of vehicle flows in and out VSAs. To achieve this, some new communications will be introduced to the simulation.
2. Prediction: To decide the vehicles movement in the mobility model, the prediction algorithm should be developed.
3. Analyzing result: To validate the prediction whether succeed in a scientific way, we will introduce the correlation formula to evaluate the different results in different experiments. Correlation indicates the strength and direction of a linear

relationship between two random variables. The correlation will be automatically calculated in a database.

The project work flow chart is presented as below.

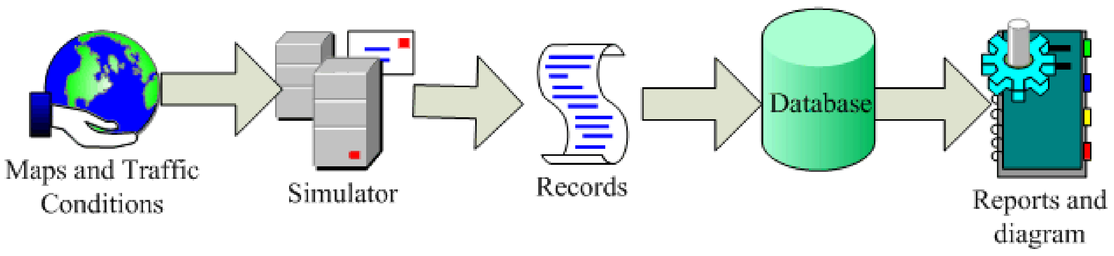


Diagram 1-A: The whole project work plan

This diagram shows how our project works like. The maps and Traffic Conditions are the Mobility models which we used. The simulator is Jist and Swans. The records and database describes in our database design. The reports and diagram is the result of our project.

In the thesis, firstly we run the mobility model on the simulator, and then through implementing the prediction algorithm to calculate and record the prediction result. Secondly, we import the result to the database to do the statistic work. In the end, we analyze the data which is export from the database.

2. Project Description

2.1. Algorithm

2.1.1. Problem definition

In the design of traffic condition prediction system, it is necessary to predict the number of vehicles are moving in the given area at the given time point in the future time. To achieve this, we should predict the drive in and out number of vehicles for that area during the given period. We record the routing information (which means the coming and target direction for the vehicles) for every vehicle that driving in the given area, then for the next time if there is a new coming vehicle drives into the given area, we select a target direction randomly in the recorded routing information as the predicting result for the coming vehicle's target direction. By doing this, we can predict the number of vehicles in the given area at a future time point that after the given period.

In this project, we separate the traffic map into several pre-defined VSAs, and each VSA is a self-stabilizing autonomous area with an elected leader. The leader is used to record the routing information and predict the vehicle's target driving direction. We model the traffic prediction as following, first, when a vehicle drives into a VSA region, the leader of the region will record the from VSA of the coming vehicle as an item of its routing information list through the message passing. Second, the leader search its routing information list, which records the recent pass by vehicles' routing information, in order to find an item that has the same coming VSA with the new coming vehicle. If there is such an item, the leader will consider the target direction, which had been recorded in the finding item, as the target direction of the new coming vehicle; else, the leader will not do the prediction. If the leader finds several items that have the same coming direction, then random function would be active to select one item. After that, the leader broadcast its prediction result to notice the VSAs which are surrounding itself. Last, the leader will complete the item of routing information by using the message that has been sent when the vehicle drive out the VSA.

For example, there is a vehicle V depart from the VSA A to the VSA B, and it will move to the VSA C at some time in the future. VSA A, B and C are the neighboring VSAs. The vehicle V says "hello and I am come from A" through message passing when the V comes into the VSA B, the leader of B will add its routing information list with a new item which include two sub-items, one is the V's coming direction A and the other is V's target direction but leave it blank. Then, the leader of B search its routing information list in order to find an item which has the coming direction is A. If there is such item, the leader considers the target direction which has stored in the item, as the prediction result. The leader broadcast its prediction result to notice the VSAs which surround it. All leaders will ignore this message except the leader of the VSA which is mentioned in the prediction result. The scenario between the V and the leader of C will be the same as we mentioned between the V and the leader of B when the vehicle V flow out the VSA B and flow into the VSA C. Also, the vehicle will broadcast its current position when the VSA has been changed. The message will be received by B's leader at the end, and then the leader will use V's current VSA instead of the blank target direction in its routing information.

2.1.2. Box

In our traffic forecasting system, we collect information in micro level and aggregate them hierarchically then make macro level prediction. The basic unit in micro level is a VSA region. The macro level contains an aggregation of VSAs' regions.

Box is either a region of a VSA or an aggregation of VSAs' regions. Considering aggregations of regions, AR = R_1, R_2..., R_k, in which the bounding and bounded boxes of AR are the same.

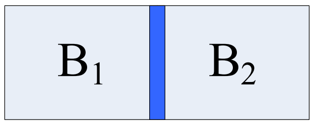


Diagram 2-A Two Neighboring Boxes

Two neighboring boxes and share edge (dark edge)

Diagram 2-A describes that, a region is a pair of locations: start-location and end-location. Meanwhile, it could also be described as a box. If the two boxes share an edge, then we say that the box B_1 and B_2 are neighboring.

A region has four coordinate attributes, two of them are the start and end coordinates in X axis, and the other two are the start and end coordinates in Y axis. The Region also has other three attributes: a pre-defined index, and two coordinates of the index which express the locations in the map of VSAs.

```
Edge
{
//properties
Start, //the begin point at x axis, the begin point at y axis
End //the end point at x axis, the end point at y axis
//functions
Public Boolean isShared (Edge e);      // is it a shared edge?
Public Boolean isExtended (Edge e);      //is it an extended edge?
}
Box
{
//properties
Start, //the begin point at x axis, the begin point at y axis
End, //the begin point at x axis, the begin point at y axis
Index, ix, iy      //the index and relate coordinates of box
List edges      //box's edges
List flows;      //the flows that passing this box
Time-stamp, // the time-span for getting these information
}
```

Our forecasting algorithm considers two neighboring boxes. It helps us to forecast traffic jam in macro level.

2.1.3.Flow

A flow can be considered as a car stream, which is a vector structure, describes vehicles passing the box. *Volume* is the number of vehicles in the flow. The *from* attribute indicate the Box that the vehicles come from. It is the start point of flow's vector. The *past* is the time which corresponding to the *from*. The *to* attribute indicate the Box that the vehicles currently stay. It is the end point of the vector. The time corresponded to the *to* is *now.*

The pseudo codes list below is the description of flow structure.

```
Flow
{
//properties
Enter-edge,     //flow's enter edge for a box
Exit-edge       //flow's exit edge for a box
Volume      //number of vehicles
//functions
Public void concatenate(Flow f); // concatenate rule
Public void merge(flow f); //merge rule
}
```

2.1.4.Zoom Out

Considering the flow data structure is based on the box definition, and it can only describe vehicle streams passing a single box. For analyzing flows in a wider and longer area (more than one box), two rules are introduced.

1. Flow concatenate: analyzing a longer area by link flows
2. Flow merge: analyzing a wider area by merging several flows

Zoom Out:

- ✓ Input: Boxes of B_1 and B_2
- ✓ Output: Box B_0
- ✓ Precondition: B_1 and B_2 are neighboring
- ✓ Action:
  1) Concatenate
  2) Merge
- ✓ Post-condition: the edges of flows of B_0 are the edges of B_0

In zoom-out calculation, we join flows from two neighboring boxes to the flows of a single united new box. And it will help us to predict traffic condition in macro level.

Firstly, aggregate the flows of the different VSAs' regions in a hierarchical manner. In order to do that, we unite regions. The decisions about how to calculate the presentation of the flows is done according to two actions that are presented below: Concatenation and merge. We apply both actions in every step of the aggregation.

2.1.5. Flow Concatenate

The following description assumes that, cars pass one region to another are under the flow concatenation and flow merge rules. Concatenation rule is explained in the following part, and the merge rule will be presented in part 2.1.6.

For analyzing flows in a longer area, the flow concatenate rule is introduced in our algorithm. This rule is used to concatenate two flows as a single flow. By doing this, we can generate a new flow which can cross more than two Boxes. Box has a pre-defined index and its coordinates(x and y), so the distance between two boxes can be calculated by $\sqrt{(X^2+Y^2)}$. With the purpose of introducing concatenation rule is to make a prediction in a longer distance.

Flow concatenation is used for predicting the flow which passes several Boxes. The concatenation rule considers two neighboring boxes, B_1 and B_2, and presents their linked flow in the box B_0. Under this rule, we only focus on the flows containing the shared edge. The edge colored dark blue in the figure below is the shared edge B_0.

- ✓ Input: Boxes of B_1 , B_2 and Flows
- ✓ Output: Box B_0 and Flows
- ✓ Precondition: B_1 and B_2 are neighboring
- ✓ Action: Concatenate
- ✓ Post-condition: the distance of output flows of B_0 are one bigger than input flows

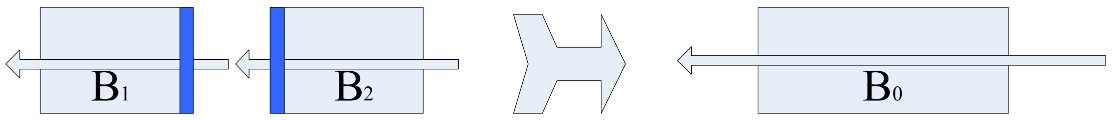


Diagram 2–B Concatenation of Flows

If the shared edge is the flow of B_1's enters edge and flow of B_2's exits edge, and the vehicle is the same in the flow of $B_1$and B_2, we concatenate the two flows as a new flow B_0 whose enter edge is B_2's enter edge and exit edge is B_1's exit edge. The volume of B_0 is equal to the volume of B_1 and B_2.

```
1    Box Concatenate{
2    If B_1 and B_2 have shared edge
3    Then B's Start = the smaller one in B_1's start and B_2's start;
4           & B's End = the bigger one in B_1's end and B_2's end ;
5           & flows = B_1's   flow +B_2's flow
6    B.flows = linkFlows(flows, sharedEdge(B_1, B_2));
7    Return B;     }
8    List linkFlows(flows, sharedEdge){
9    While (flows list are not empty)
10   {put the flows in two lists: Enter and Exit
11   If (flow's enterEdge == sharedEdge)
12         Enters<-Enter ∪ flow and delete this flow in the flows list;
13  Else If (flow.exitEdge == sharedEdge)
14         Exit<-Exit ∪ flow and delete this flow in the flows list;}
15   While (Enter list is not empty){
16     While (Exit list is not empty){
17          if flowExit match flowEnter
17           then {new flow f's exit = Enter's exit;
18                          f's enter = Exit's enter
19                    f's volume = Exit's volume;
20                    put f in List fs;}
21          f<-f∪ flow and delte this flow in the Enter and Exit list, break;}}
22    Return fs;}}
```

The line 2 is a judgment for whether two boxes are the neighboring box. From line 3 to 5 set the start and end point for new box and calculate the number of flows in the new box. Line 6 is a method invocation to calculate the flow attribute for the new box B. Line 9 is used to judge is there any item remain in the flows list. If list is not empty, from line 10 to 14 will do a loop to separate flows by shared edge. Line 11 to 12 put the flows into the Enter list if flow's enterEdge is the shared edge and delete the flow in the flows list. Line 13 to 14 put the flows into the Exit list if flow's exitEdge is the shared edge and delete the flow in the flows list. From line 15 to 19 is a double loop to find match flows that can be concatenated, set the new flow's enter and exit edge, and the volume of flow. Break the inner loop to start a new round of matching.

For example, there are two flows which are generated by a vehicle passing by VSA B_1 and B_2 in diagram 2-B. One flow is passing by B_2 and exit B_2 through B_1 and B_2's shared edge, another flow is enter B_1 through B_1 and B's shared edge and passing by B_1. In another word, because of the B_1 and B_2 are neighboring boxes, then the vehicle drives into B_1 while it drives out B_2. We can concatenate these two flows into a new flow which the enter edge is B_2's enter and the exit edge is B_1's exit.

2.1.6.Flow Merge

For analyzing flows in a wider area, the flow merge rule is introduced in our algorithm. This rule is used to merge small flows to branches of flows. By doing this, the trends of car stream can be found, but some details are ignored.

- ✓ Input: Box B_0 and Flows
- ✓ Output: Box B_0 and Flows
- ✓ Precondition: Flows are passing B_0
- ✓ Action: Merge
- ✓ Post-condition: the sum number of B_0 merged equals the sum number of input flows

In this rule, we focus on the flows containing the same flow in or the same flow out edges. The edges colored dark gray in the figure below.

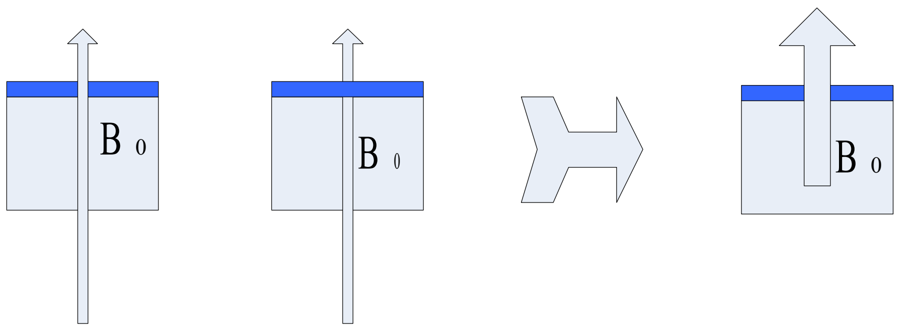


Diagram 2–C Merging of Flows

Considering the diagram 2-C, if only two flows exit B_0 at the given edge (dark edge) during a given period (This given period will be work as the prediction period in our implementation chapter), and flows exit B_0 at different time point, we join them together as a single flow which the volume of the new flow as the summation of the old two flows. The new flow will be considered as the new flow out for this given edge during the given period. This is the merge rule for flow out vehicles, the flow in vehicles merge rule are the same statement as flow out.

```
Box merge (Box B) {
1       while (list B's flow is not empty){
2         If (f_1 and f_2 have same enter and exit edge){
3             f_new's enterEdge =Enter edge & f_new's volume <- f_1 + f_2
4             F_new's exitEdge = edge & delete the flow from the flows;}}
5   B.flows <- f_new;
6   Return B;}
```

From line 1 to 4 is a double loop that we search all flow for every edge of a box, in order to find the flows, which have the same enter and exit edges, that we merge them. At line 4 we reset the enter edges, exit edge and the volume of flow. At line 5 we restore B's flows.

For example, there are two flows which are generated by two different vehicles at two different time in diagram 2-C. Considering the same enter and exit edges for these two flows, we can merge them into a new flow which the volume is the summation of the two flows. We say that the new flow is the flow of B_0 in given period time.

2.1.7. Prediction Formula

Our prediction is based on the volumes of flows (x_{in} and x_{out}), which it is means the summation of flow in and out during a given period P (here given period will be implemented as the length of prediction time), and base number x_{base} the volume of vehicles in a box at a certain time T. The formula computes the volume of vehicles in a box at time (T+P). The P is a constant number which could be any numbers (0, 10, and 20…e.g.). The x_{in} is the volume of coming in vehicles from time T to T+P for the box. The x_{out} is the volume of going out vehicles from time T to T+P for the box.

The predicted number is:

$$f(x) = x_{base} + x_{in} - x_{out}$$

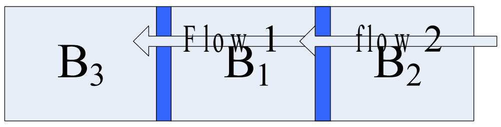
Formula 2-A formula of prediction

The flowing description is an explanation for how does the traffic prediction has done.

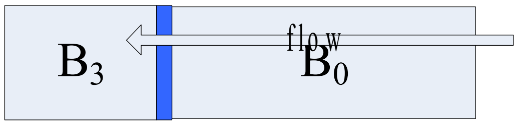
There are three pre-conditions for the algorithm, we assume that

1. mobile nodes can communication with each other
2. every mobile node know its location
3. geography map is plotted out into several pre-define regional self-autonomy

There are three boxes B_1, B_2 and B_3, which are sharing two shared edges (dark blue in the fig below). The flow1 is crossing $B_1$and tend to flow in B_3. The flow2 is crossing B_2 and tend to flow in B_1.



We can use concatenation rule with $B_1$and B_2 to construct a new box B_0

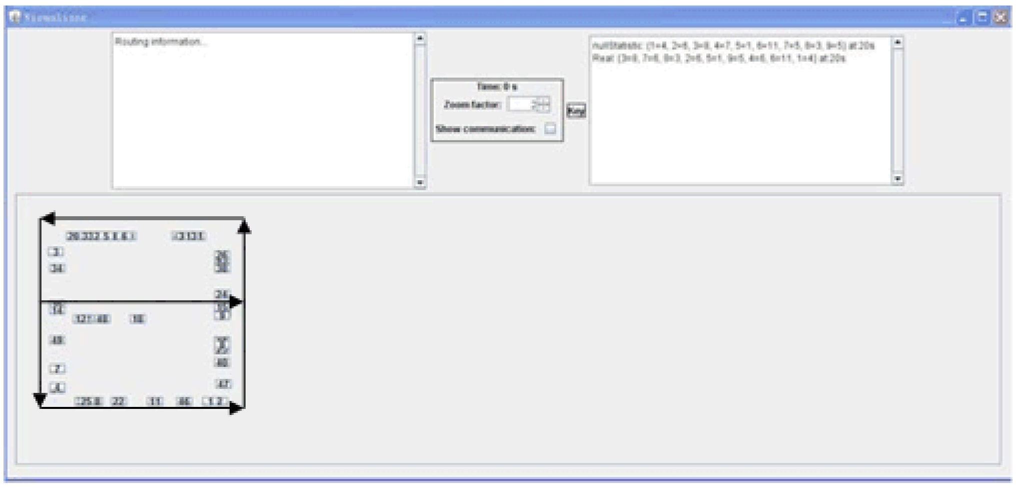


With the timestamp of B_0 we can know how many mobile nodes will flow into the box B_3 in the certain time by crossing the shared edge between B_0 and B_3. Then we use the merge rule to merge the flows that passing the shared edge to get the x_{in} which we mentioned in the prediction formula. Repeat these steps to get the flows which passing the other three edges for B_3, and then we can get all x_{in} and x_{out} of B_3. At the end, the prediction can be done by using the prediction formula we has mentioned in chapter 2.1.6.

2.2. Mobility Models

2.2.1. Double Loop Mobility Model

During the developing of our prediction model, a simple mobility model has been applying to simulate some idea traffic condition. It is double loop mobility model.



```
nullStatistic (1=4, 2=6, 3=8, 4=7, 5=1, 6=11, 7=5, 8=3, 9=5) at 20s
Real (3=8, 7=8, 9=3, 2=6, 5=1, 9=5, 4=6, 6=11, 1=4) at 20s
```

Diagram 2‑D: Double Loop Mobility Model in the graphic interface

In the model, we implement two virtual paths: outer loop and inner loop. 66% vehicles circle the outer loop, while the other 33% follow the inner loop. The Double Loop Mobility Model helps us find bugs in our system.

1.1.1.Street Random Waypoint

Street Random Waypoint (STRAW) mobility model is developed by AquaLab Project. Its aim is to implement and deploy network protocols and services for vehicular ad-hoc networks (VANETs) on simulation for evaluation. The most attractive advantage of Street Random Waypoint mobility model is that, this mobility model can be considered as a realistic vehicular mobility model [6]. The model ensures conclusion drawn from previous experiments will be implemented in real deployments.

Unlike many other mobile ad-hoc environments, the most different part of VANETs is that the movement of nodes constraint to streets. And they are usually separated by buildings, trees, or other objects. Such objects increase distance between nodes and may reduce overall signal strength for each node. The STRAW mobility model implements streets which are defined by real map data from U.S. Census Bureau [6].

The performance of wireless network protocols in urban environments is dramatically different from that in the open-field Random Waypoint model. Moreover, the type of urban environment can significantly impact the performance of a protocol [6].

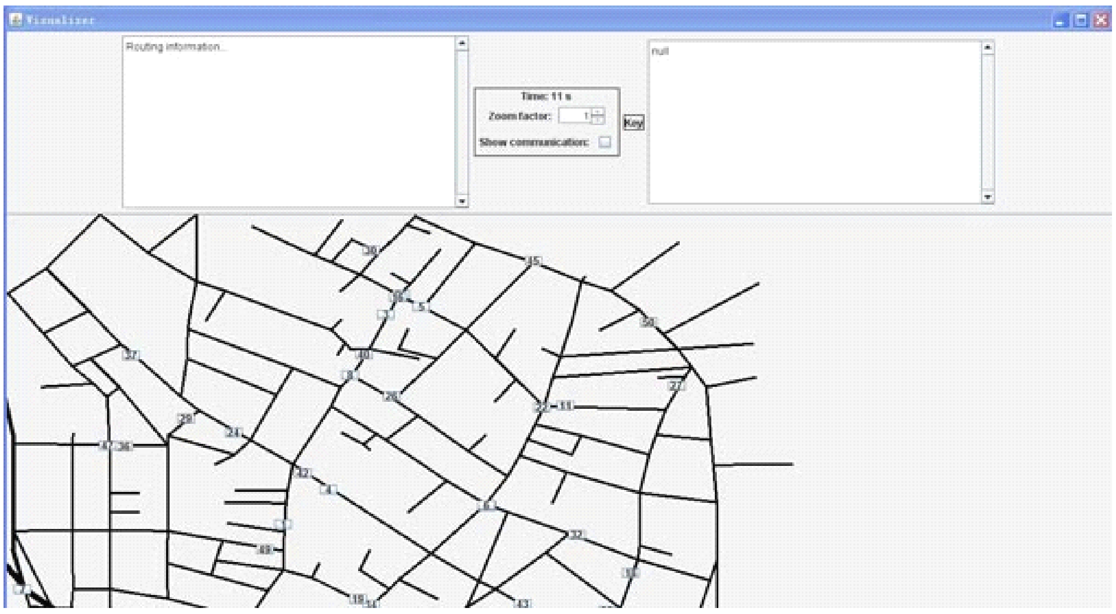


Diagram 2-E: STRAW in the graphic interface

Because STRAW is the most matured and realistic mobility model for VANETs [6], we choose it as our main mobility model. All the results we got are based on this mobility model.

2.3. Implementation

2.3.1. Time Slice

A time slice is used to describe the VSA's life time which as long as 5000 milliseconds (ms) for every round. All nodes send join or restart join message to elect a leader for the VSA which they are stay in. The leader of VSA is elected at 210 milliseconds by the received join and restart join messages, and it will die at 4800 milliseconds. Also at that moment, leaders of VSAs send message to report the information of their own region. All nodes send GPS messages for every 1000s milliseconds, if their geographic locations changed. A new time slice begins, after time out of 5000 milliseconds.

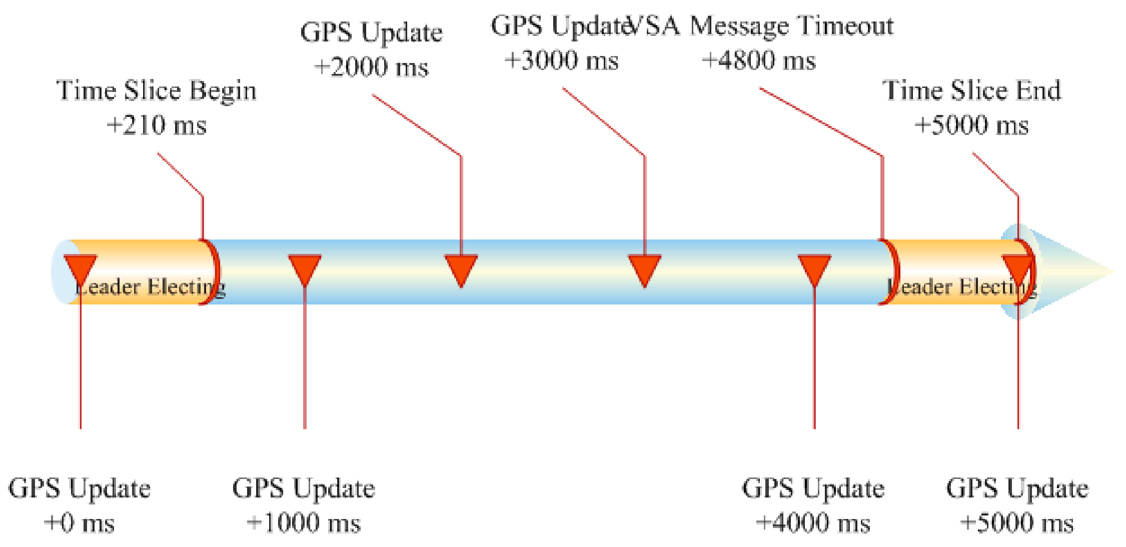


Diagram 2–F: a time slice using in VSA

2.3.2. Messages

Flow

A flow is sending by a vehicle node, represent as a message which contains such fields:

Flow Message						
ID	From	To	Passed	Identical Time	Time Span	Time Stamp

Table 2–A Structure of Flow Message

Graph of track

1. Node *ID*

   The unique identical of a vehicle

2. *From* region

   The region where the node come from

3. *To* region

   The region where the node are currently locating

4. *Passed* region

The region where the node originally come from

5. *Identical Time*

   The moment when the vehicle identifies the VSA region changed.

6. *Time Span*

   The duration of the vehicle spent for passing from the *From* region to the *To* region.

7. *Time Stamp*

   The moment when the vehicle send the flow message

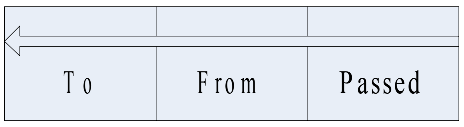


Diagram 2-G Flow

When receiver receive a message, it will firstly check the *From* and *To* fields in the message. If the *From* filed equals the region that receiver belongs to, the receiver will consider that flow as a going out flow (flow out). If the *To* field in this message equal to the region that receiver belong to, the receiver will consider that flow as a coming flow (flow in). The flow message will be sent three times in every time slice, the sending time has been marked below (500, 2500 and 4500 milliseconds).

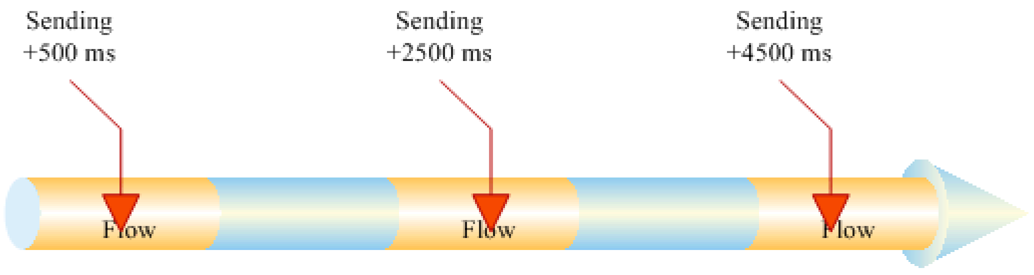


Diagram 2-H Time Slot of Sending Flow Message

Virtual Flow

The prediction represents by a data structure virtual flow. It is also a message. It contains such fields:

Virtual Flow Message					
ID	From	To	Expected Time	Time Stamp	Count

Table 2–B Structure of virtual flow message

1. Node *ID*:
   The unique identical of a vehicle
2. *From* region:
   The VSA which made this prediction (also called predicting VSA).
3. *To* region:
   The VSA which we predict the node would go to (also called predicted VSA).
4. *Expected Time*:
   The time when the vehicle will reach the *To* region.
5. *Time Stamp*:
   The time when the message has been generate
6. *Count*:
   the remains distance for further prediction

The virtual flow message will be sent twice in every time slice, the sending time has been marked below (1500 and 3500 milliseconds).

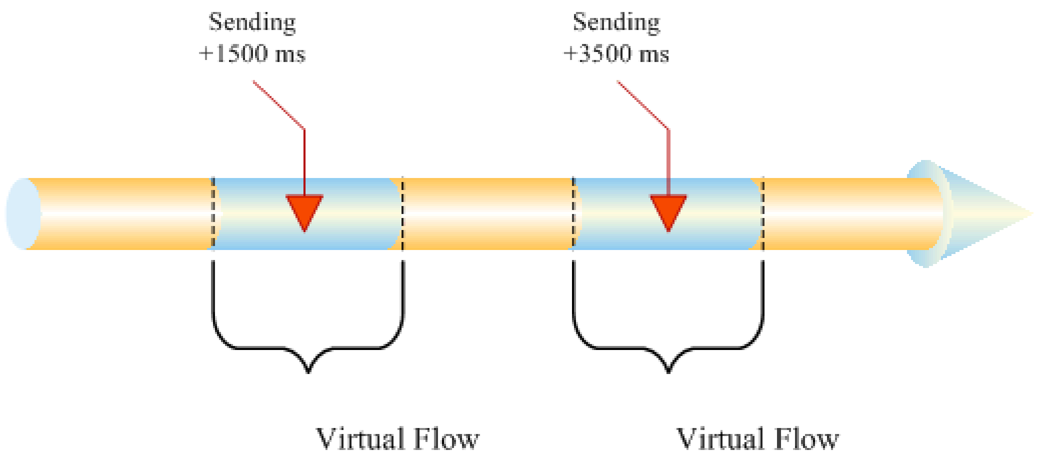


Diagram 2–I Time Slot of Sending Virtual Flow

2.3.3. Box

Based on the definition in chapter 2.1, the box is either a region of a VSA or an aggregation of VSAs' regions. The box implementation here is considered as a VSA region.

Every VSA region has one leader to maintain the membership of its region. All other nodes registered as members in leader list. So that the VSA knows how many vehicles are in its region.

The membership is represented by a map data structure:

The key of the map is the vehicle node *ID*, while the value is timestamp. The timestamp is the time when a vehicle node being registered or updated as a member.

There are three operations may update the membership map, they are:

1. Register:
   Once the leader of VSA received a flow in, it will try to register the vehicle node to its membership map. If there is no such a vehicle node existing in the membership map before, the leader of VSA will register the node as its member. If it exists, the leader of VSA will try to compare the timestamp of flow message and the timestamp of membership map, update the value of membership map as the newer one.
2. Unregister:
   Once the leader of VSA received a flow out, it will try to unregister the vehicle node from its membership map. It will simply remove such a node from the membership map.
3. Timeout:
   Because it is possible that, the leader of VSA cannot receive the flow out, we implement time out function to refresh the membership map. So that we can keep membership more accurately. The way to do that is to dump the old key-value set, if it is older than the current time minus the time out, we set in constant class.

2.3.4. Flow Prediction

Our implementation predicts flow by generating virtual flow from flow message and roads information. The road information is maintained in each box by the leader and the guard of VSA.

Road

The road is the set of information we collect from flow messages, which composes by path information and functions for predicting traffic.

The road represents by a map data structure. The key is the From Region, while the value is a list. The list contains path. The path with the same From Region will be stored together. Even the To Region is the same, it is also stored. Because in the following routing algorithm, the more times the To Region appear in the list of paths, the easier the paths could be selected by Random Function Method.

Diagram 2–J Road Map

The path is a data structure. It contains such fields:

Path		
From	To	Time Span

Table 2–C Structure of Path

1. *From* region

   The enter region of that path.
2. *To* region
   the exit region of that path
3. Time *span*

   The time cost on that path. It represents the length divided speed of that path.

The single path information is collected by the leader. When leader receive a flow out message, it will record the *Passed* region in the message as the *From* region in the path, record the *To* region in the message as the *To* region in the path, and record the time span in the message as the *Time Span* in the path. Once the leader creates a path according to the flow out message, it will put that path to the road map. The road map gathers such information by the *To* region of path.

Routing

The routing function is to pre-select a path for vehicle node will go to. It means forecasting.

The leader will make a prediction once it receives a flow message. The prediction contains the information of where the vehicles going and the time of arriving. There are few steps which could impact the prediction information:

Once a leader receives a flow in message send by a vehicle, it will try to find out whether the from-region is the existed key in its road map. If it does not, there is no prediction make for the vehicle. If it does, a random index of that list will be generated, so that we can select a path from road map.

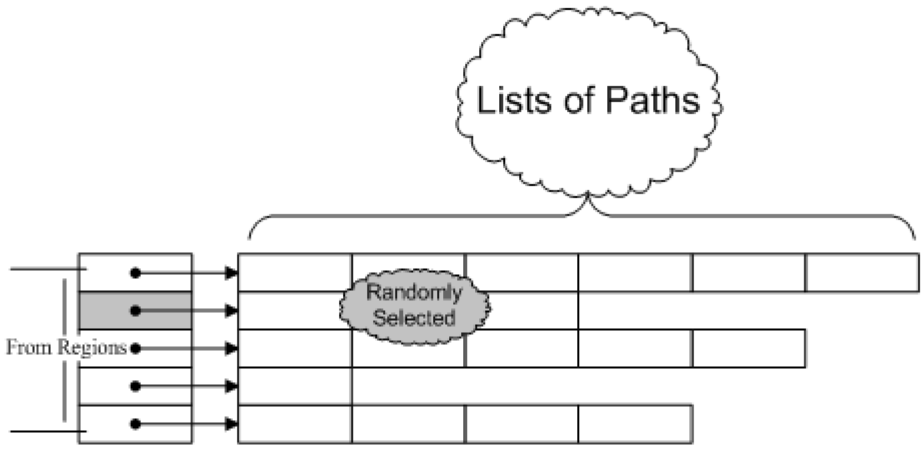


Diagram 2–K Structure of road

Once a path has been selected, the prediction makes at the same time.

Cache

For keeping the consistent of our prediction and the speeding up the performance of the system, we introduce the cache map.

Since the leader of VSA receives the same flow message more than once, it is possible that the leader makes more than two different predictions for the same vehicle node, which possibly violates the consistent of the prediction. And it will confuse other VSAs by making the prediction that one vehicle comes into two different VSA at the same time in future.

The key of the cache map is the flow message, while the value of this map is the virtual flow which had been predicted by the leader. Once the leader receives a flow message, it will firstly try to find out whether this flow message exists in the cache map. If it exists, the leader will no longer make a prediction for that flow message, but return the virtual flow which is related with the key.

Cache	
Hash Code of Flow	Virtual Flow

Table 2–D structure of cache

2.3.5. Flow Calculation

In the flow merging algorithm: if flows have the same enter edge or exit edge, we join them together as a single flow which the volume of the new flow as the summation of the old flow volumes. The new flow will be considered as the new flow in or flow out for this edge. The leader of VSA will do the flow merging.

The data structure used to implement flow merging is Map. An map (also associative container, mapping, hash, dictionary, finite map, and in query-processing an index or index file) is an abstract data type composed of a collection of unique keys and a collection of values, where each key is associated with one value (or set of values). The relationship between a key and its value (or set of values) is called mapping or binding.

In our implemented map, we use the Predicted Region to represent key and exit edge. And the key binds a set of values. The values are the received virtual flow messages and have the same predicted region. So we merged a number of virtual flows to a certain number of flows. The number is the as the same as the different kinds of predicted regions. By doing this, we get new flow which the predicting region is region of VSA leader, the predicted region is the key, and the volume is the size of a set of values mapping to the key.

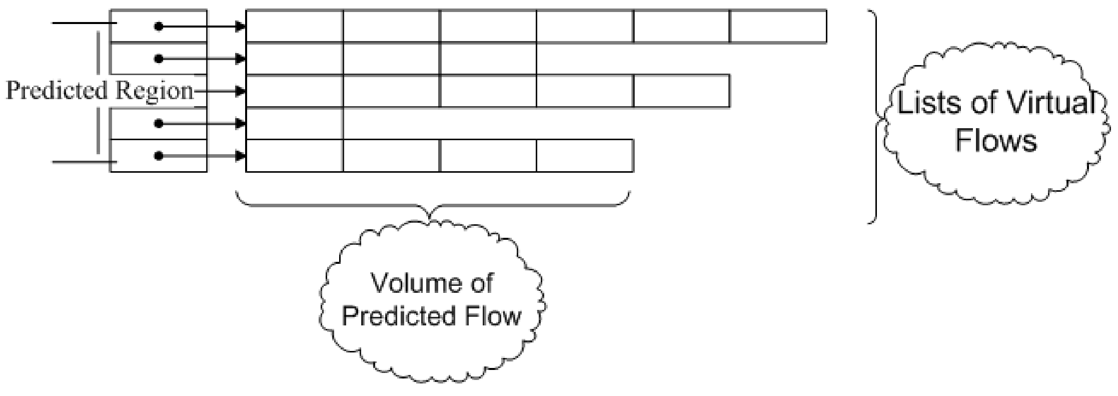


Diagram 2–L structure of flow merging map

Flow concatenation is used for predicting the flow which passes several VSAs. The concatenation rule considers two neighboring boxes, B_1 and B_2, and presents their linked flow in the box B_0.

In the flow concatenation algorithm: if the shared edge is the flow of B_1's enters edge and flow of B_2's exits edge, and the vehicle is the same in the flow of $B_1$and B_2, we concatenate the two flows as a new flow B_0 whose enter edge is B_2's enter edge and exit edge is B_1's exit edge. The volume of B_0 is equal to the volume of B_1 and B_2.

In our implementation: the system generate virtual flow message by treating received flow message almost like the same of flow message.

But it has one difference, and we explain the difference at two steps:

1. If the count field of virtual flow message is 0, the system ignores and drop received virtual flow message. If the count is bigger than 0, the system will do the step 2.
2. The system generates a new virtual flow message by the received virtual flow mes

message. The id, from, time stamp and to fields will be treated like the same of flow message. The expected time field will be treated like identical time of flow message. The passed field will be treated as null. The system will skip the road part of flow prediction but keep doing the routing and cache part of prediction.

3. At last, set the count of new virtual flow message 1 smaller than the received virtual flow message.

3.3.7. Recording

Before the end of a time slice, all the information we gathered will be filled into five different tables. They are represented by five different CSV (Comma Separated Value) file.

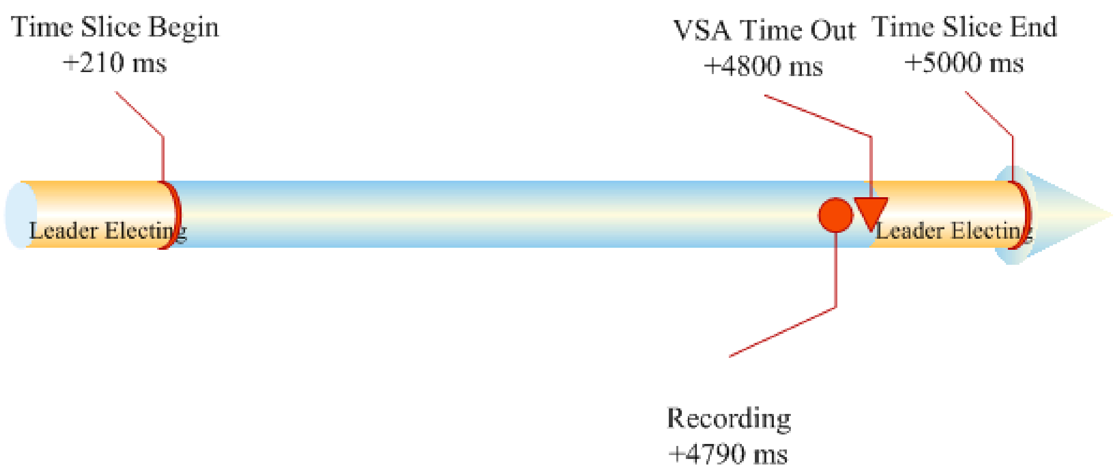


Diagram 2–M: The moment of Recording

The details of format for five CSV fields are as following:

1. Prediction Table:
   This table records all the predictions made by the leader of VSA. Each line represents a single perdition. The predicting VSA column records the index of VSA who made such a prediction. The Now column records the time when the VSA made that prediction. The Predicted VSA column is the target VSA of the prediction. The Future column is the predicted time. The Num column is the predicted number of vehicles of that prediction.

File Name: Pre.csv				
Predicting VSA	Now	Predicted VSA	Future	Num

Table 2‑F structure of prediction file

2. Result Table:
   This table records the result of real number of vehicles in each VSA region at certain time point. It is calculated by the simulator. It is used to measure our prediction between the real situations.
   The VSA column is the index of VSA. The time is the time when we got such a result. The Num column is the actual number of vehicles in the VSA.

File Name: result.csv		
VSA	Time	Num

Table 2‑G structure of result file

3. Coordinate Table:
   This table records the VSAs' indexes and their relate coordinates. It is recorded by simulator, once VSAs were initiated.

File Name: region.csv		
VSA Index	VSA X	VSA Y

Table 2‑H structure of region file

4. Real Flow Table:
   This table records all flows collected by the leader of VSA. Each line represents a single flow. The From VSA column records the index of VSA who made the flow statistics. The Start Time column records the begin time when the flow begins. The To VSA column is the target VSA of the flow. The End Time column is the end time of the flow. The Volume column is the number of vehicles in that flow.

File Name: f.csv				
From VSA	Start Time	To VSA	End Time	Volume

Table 2‑I structure of flow file

5. Virtual Flow Table:
   This table recorded all the virtual flow predictions made by the leader of VSA. Each line represents a single perdition. The predicting VSA column records the index of VSA who made such a prediction. The Now column records the time when the VSA made that prediction. The Predicted VSA column is the target VSA of the prediction. The Future column is the predicted time. The Volume column is t

the number of vehicles is predicted in that virtual flow.

File Name: vf.csv				
Predicting VSA	Now	Predicted VSA	Future	Volume

Table 2-J structure of virtual flow

2.4. Database Design

2.4.1. Entity Relationship

Considering the large quantity of calculation and reusability, the statistic work has been processed by Access database. The below ER-diagram is the relationship between the tables.

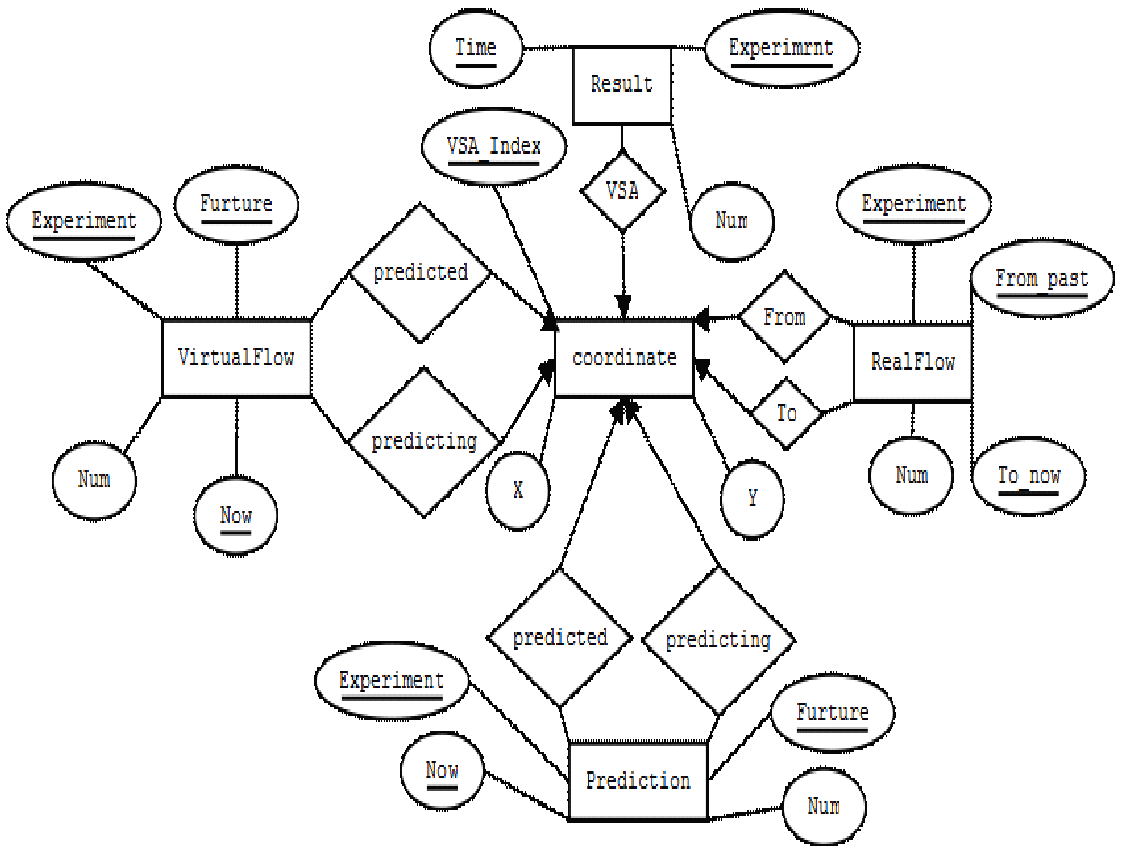


Diagram 2-N: Entity Relationship Map

2.4.2.Named Queries

We get three groups of number from the simulation: reality numbers, real numbers and prediction numbers. Reality number: the number of vehicles calculates by simulator at now time. Real number: the number of vehicles calculates by a VSA at a passed time point, and the number of flows between the passed time point and now. Predicted number: the number of vehicles calculates by a VSA at now time, and the number of virtual flows between the now time point and a future time point. In order to get the correlation between the prediction numbers of vehicles, the reality number of vehicles and the real number of vehicles in VAS regions, we use the following queries.

1. Fixed_flowOut:
   It is a named query calculating all numbers of vehicles flow out the particular VSA region during the certain prediction period. The query is generated from the RealFlow table.
2. Fixed_flow In:
   It is a named query calculating all numbers of vehicles flow into the particular VSA region during the certain prediction period. The query generate from the RealFlow table.
3. Fixed_flowInflowOut:
   It is a named query calculating the sum numbers of vehicles flow out and flow in the particular VSA region during the certain prediction period. The number maybe a negative number, because of the flow out number is bigger than the flow in number during that period. The query generate from the Fixed_flowOut and Fixed_flowIn queries.
4. Fixed_num:
   It is a named query calculating the number of vehicles in the particular VSA region in the certain time point with the certain prediction period. The query generates from the Fixed_flowIn_flowOut query and Result table.
5. Pred:
   It is a named query calculating the sum of number of vehicles in the particular VSA region in the certain time point with the certain prediction period. The number maybe a negative number, because of the flow out number is bigger than the flow in number during that period. The query generates from the Prediction table.
6. all_num:
   It is a named query comparing the prediction number, real number and reality number of vehicles in the particular VSA region in the certain time point with the certain prediction period. The query generates from the Result table, Pred query and Fixed_num query.
7. perAvg:

It is a named query calculating the average and standard deviation of prediction number of vehicles group by different period, and the average and standard deviation of reality number of vehicles also group by different period. The query generates from the all_num query.

8. perCor:
   It is a named query calculating the correlation between the prediction number and reality number in different period. The query generates from perAvg.
9. fixAvg:
   It is a named query calculating the average and standard deviation of real number flows of vehicles group by different period, and the average and standard deviation of reality number of vehicles also group by different period. The query generates from the all_num query.
10. fixCor:
    It is a named query calculating the correlation between the real number flows and reality number in different period. The query generates from fixAvg.
11. dis0:
    It is a named query comparing the prediction number and real number of vehicles in the particular VSA region in the certain time point, certain prediction period, and certain distance 1. The query generates from Prediction, Result, and coordinate table.
12. dis1:
    The same as dis0, but with the distance not bigger than 1.
13. dis2:
    The same as dis0, but with the distance not bigger than 2.
14. dis3:
    The same as dis1, but with the distance not bigger than 3.
15. dis0AVG:
    It is a named query calculating the average and standard deviation of prediction number of vehicles group by different period, and the average and standard deviation of reality number of vehicles also group by different period. The query generates from the dis0 query.
16. dis1AVG:
    The same as dis0AVG but generate from dis1 query.
17. dis2AVG:
    The same as dis0AVG but generate from dis2 query.
18. dis3AVG:
    The same as dis0AVG but generate from dis3 query.
19. dis0Cor:
    It is a named query calculating the correlation between the prediction number and

reality number in different period. The query generates from dis0AVG.

20. dis1Cor:
    The same as dis0Cor but generate from dis1AVG query.
21. dis2Cor:
    The same as dis0Cor but generate from dis2AVG query.
22. dis3Cor:
    The same as dis0Cor but generate from dis3AVG query.
23. disCorComp:
    Compare the correlation with different distance, group by period.

2.4.3. Correlation

Correlation indicates the strength and direction of a linear relationship between two random variables. The Correlation formulate has been used in statistical broad in this thesis to calculate how close between two groups of variables, they are Real Flow and Virtual Flow, Prediction number and Reality number, Prediction number and Real number. It refers to the departure of two variables in a group from independence, measuring the degree of correlation

$$\rho_{X,Y} = \frac{\mathrm{cov}(X,Y)}{\sigma_X \sigma_Y} = \frac{E((X-\mu_X)(Y-\mu_Y))}{\sigma_X \sigma_Y},$$

Formula 2–C

$$\mathrm{corr}(X,Y) = \rho_{X,Y}\ .$$

Formula 2–D

$$\rho_{X,Y} = \frac{E(XY) - E(X)E(Y)}{\sqrt{E(X^2) - E^2(X)}\ \sqrt{E(Y^2) - E^2(Y)}}.$$

Formula 2–E

The correlation is defined only if both of the standard deviations are finite and both of them are nonzero, and the correlation cannot exceed 1 in absolute value.

The result is generated by correlation formulate here is the guide line of linear relationship degree。The correlation is 1 in the case of an increasing linear relationship, -1 in the case of a decreasing linear relationship, and some value in between in all other cases, indicating the degree of linear dependence between the variables. The closer the coefficient is to either −1 or 1, the stronger the correlation between the variables. If the result is larger than zero, it is the positive correlation, which means if one variable is bigger; the other one is also bigger. If the result is smaller than zero, it is the negative correlation, which means if one variable is bigger; the other one will be smaller.

If the variables are independent then the correlation is 0, but the converse is not true because the correlation coefficient detects only linear dependencies between two variables, but maybe other kinds of correlation.

If the absolute value of result is bigger, the correlation between the numbers in a group will be much closer, and the points distribute in the map will be close by linear regression line. Complete positive correlation (result equal to one) or complete negative correlation (result equal to negative one), all the points are distributed in the linear regression line. And the points are distributed discrete to the linear regression line if the absolute number is small.

The flowing table has been separate in three different levels to describe the degree of relationship between two variables.

Correlation	Negative	Positive
Small	-0.3 to -0.1	0.1 to 0.3
Medium	-0.5 to 0.3	0.3 to 0.5
Large	-1.0 to 0.5	0.5 to 1.0

Table 2–K

First we use the correlation formulate calculate the relationship between a group of prediction result and a group of reality result

3. Results

3.1. Numerical Result

In this chapter, we shall examine three groups of correlation for analyzing short term traffic prediction: correlation between prediction and reality numbers, correlation between prediction and real numbers and correlation between reality and real numbers.

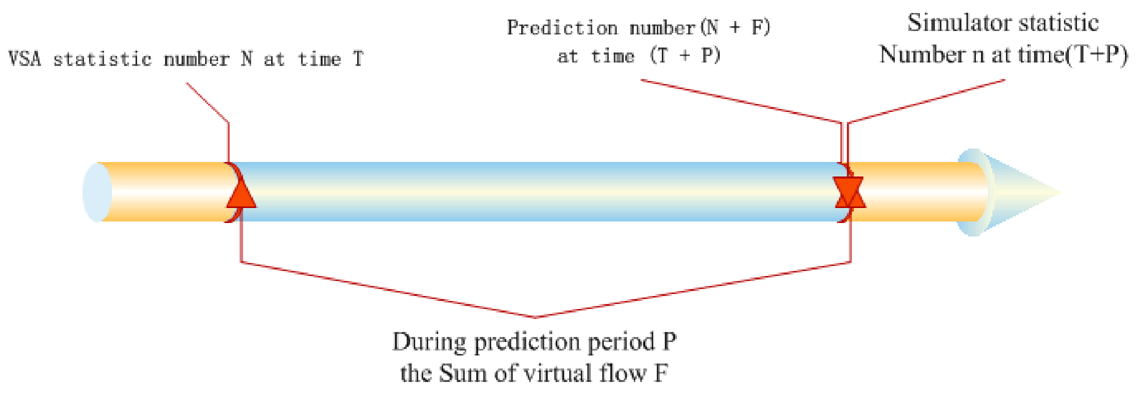

VSA statistic number N at time T
Prediction number (N + F) at time (T + P)
Simulator statistic Number n at time(T+P)
During prediction period P the Sum of virtual flow F

Diagram 3–A time of result

use this N plus the summation of virtual flow number F (flow out and flow in, predict by VSA leader) during the certain prediction period P.

Reality number equal to the statistic number n by simulator at time T+P.

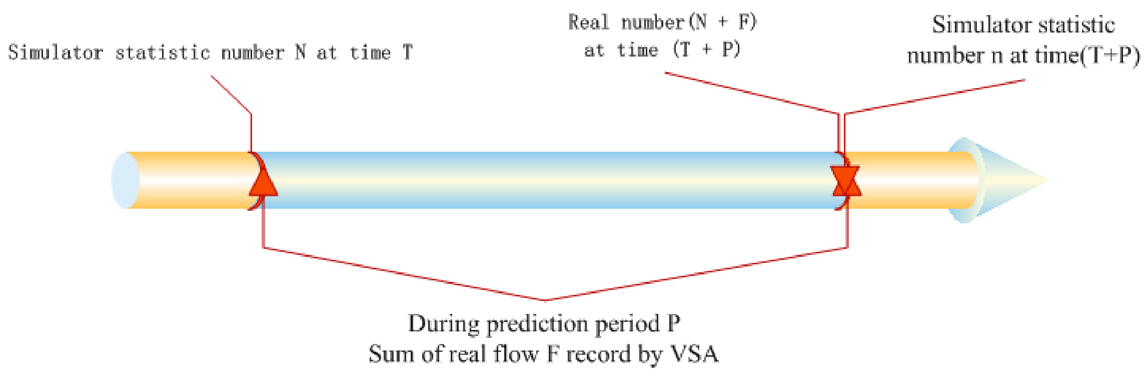

Simulator statistic number N at time T
Real number (N + F) at time (T + P)
Simulator statistic number n at time(T+P)
During prediction period P Sum of real flow F record by VSA

Diagram 3–B Recorded time period

Real number equal to the number N statistic by simulator at time T, use this N plus the summation of real flow number F (flow out and flow in, statistic by VSA leader) during the certain prediction period P.

The correlation indicates the strength of a liner relationship between two variables. So, the first group: correlation between prediction number and reality number is used to indicate whether our prediction result is close to the real number that calculate by simulator in background at any given time point. In another word, it is reflect the strength between our prediction result and the real number of vehicle. This correlation is used to validate the accuracy for our prediction. The second group: correlation between real and reality number is used to indicate the vehicle members calculate by VSA's leader whether close to the numbers that calculate by simulator in background at any given time point. This result is used to validate the accuracy of VSA's leader recording for the passing vehicles. The main problem for this correlation is that the leader may miss some vehicle records because the messages undelivered. The last, correlation between prediction number and real number is used to indicate the relationship between the prediction results and the number of passing vehicles which recorded by VSA's leader in a VSA. The aiming for this calculating is used to eliminate the effecting of the leader's wrong statistic for the correlation. It is used to validate the predicting flow in and out accuracy. Also we compare the prediction vehicle flows and the real vehicle flows in different prediction periods. The reason for doing this is find the best prediction period for our algorithm. At the end of this chapter, we compare the correlation of prediction result to find a trend of decreasing for different prediction periods with four levels of prediction distance.

The prediction can be modeled by the formula: $f(X)_{vsa} = X_{base} + X_{in} - X_{out}$, where $f(X)_{vsa}$ is the number of vehicles in the VSA after time T, X_{base} is the number of vehicles in the VSA at right now, X_{in} is the number of vehicles flow into the VSA during time T, and X_{out} is the number of vehicles flow out the VSA during the time T. By calculating the prediction flows in and out for the given VSA in a given period T, and plus the vehicle numbers at now time, then we can get the prediction number after time T.

For example, there is a vehicle V depart from the VSA A to the VSA B, and A B are the neighboring VSAs. Our prediction algorithm predict V will drive to B in some times latter, this information is sent by leader of A and received by leader of B. Leader of A consider V as the prediction X_{out}, and leader of B consider V as prediction X_{in}. After a period of given time, the scenario will repeat for several times, then both of A and B use the formula to accumulative total number of vehicles in their region.

3.1.1. Correlation between Prediction and Reality

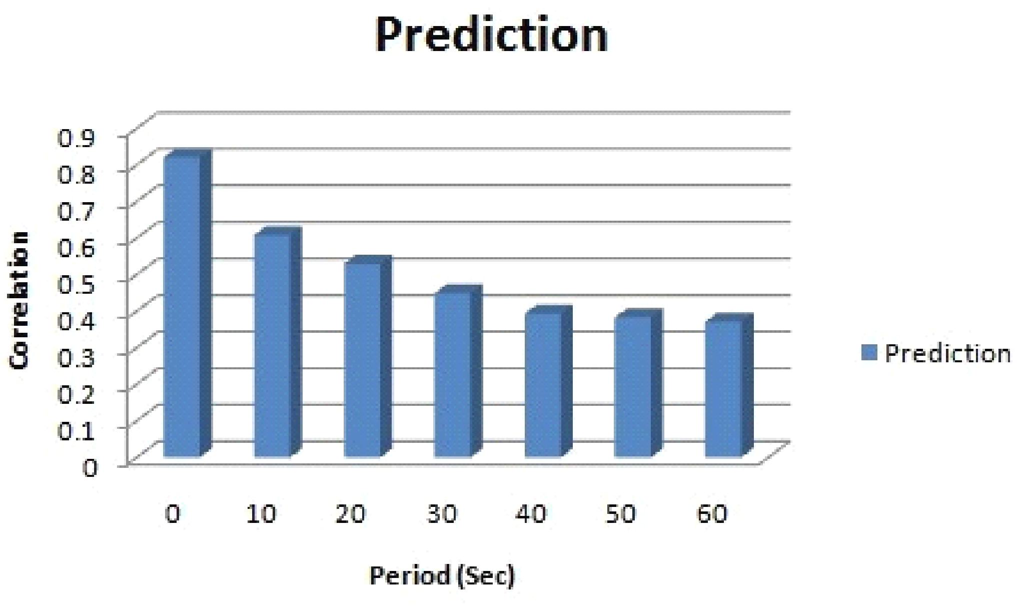


Diagram 3–C Correlation of prediction between different periods

Period	Correlation
0	0.850132
10	0.618967
20	0.570963
30	0.538576
40	0.509607
50	0.492807
60	0.473103

Table 3–A Table of Correlations

This is the correlation between reality number and prediction number vehicles with different prediction period at certain time point. From the table, it is clear reflect the correlation is decreasing by the increasing of the prediction period. The first reason for this is the accuracy of flows prediction decreased after the prediction period extend to a certain long time, this can lead to wrong number when calculate the sum of vehicles in VAS region. Prediction maybe wrong or do not predict at all with some long enough period. The second reason is VSA leader statistic members of itself wrong. The reason for this is message not received.

3.1.2. Correlations Comparing

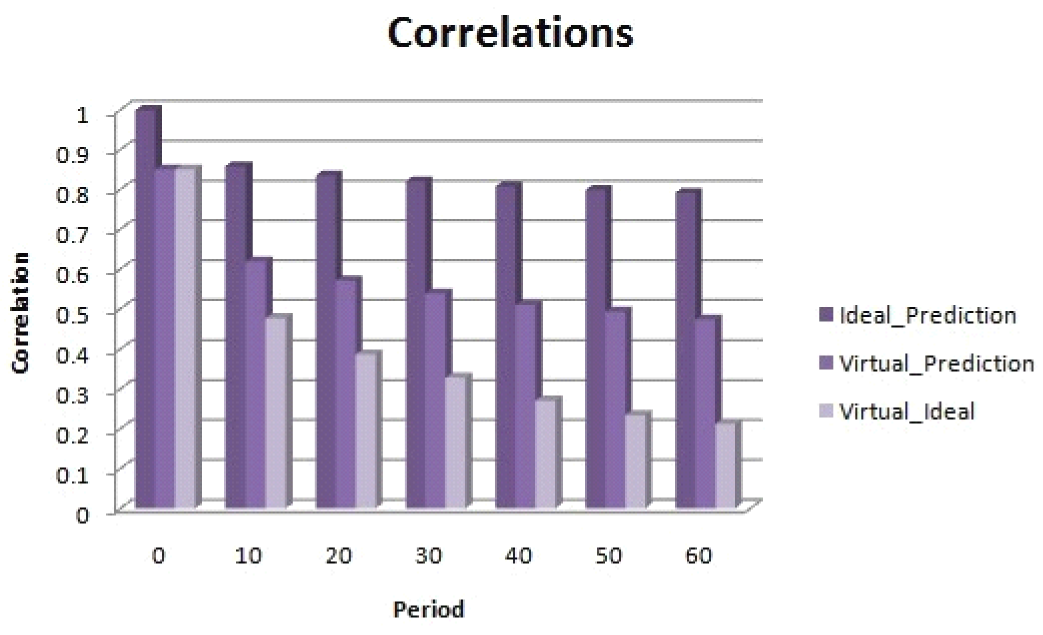


Diagram 3–D Correlations comparing

Period	Virtual_Prediction	Ideal_Prediction	Virtual_Ideal
0	0.850132	0.999162	0.850132
10	0.618967	0.858205	0.476604
20	0.570963	0.834941	0.386843
30	0.538576	0.820078	0.328544
40	0.509607	0.807629	0.270638
50	0.492807	0.798541	0.23428
60	0.473103	0.791506	0.212535

Table 3–B table of correlations comparing

This is the comparison of correlation Virtual_Prediction, Ideal_Prediction and Virtual_Ideal. Virtual_Prediction is the correlation between prediction number and reality number with different period at certain time point. Ideal_Prediction is the correlation between the real number and reality number with different period at certain time point. Virtual_Ideal is the correlation between the prediction number and real number of vehicles with different period at certain time point. The real number means that real numbers statistic by VSA at certain time point plus real flow in and flow out numbers during a certain period that before the next certain time point. Theoretically, I

Ideal_Prediction correlation should be equal to one. Because it use the real number statistic by VSA's leader and plus the real flows record by VSA's leader. This summation should be equal to the reality number in this VSA. But from the diagram, this correlation is only around 0.8. The main reason for this is the VSA messages delay and VSA's leader can not record all members and flows in it.

Theoretically, the correlation of Virtual_Ideal is between the correlation of Virtual_Prediction and Ideal_Prediction. That is because the prediction base on the VSA leader's record (record means the flows and members in this VSA region), so the prediction number should be near the real number. But here the prediction number has more relationship with reality number. So, Virtual_Ideal is the lowest correlation in three of them.

Prediction and Ideal Prediction

It is a comparison of the ideal prediction correlation and prediction correlation.

3.1.3.Flow Comparing

Variance of Real Flow and Virtual Flow

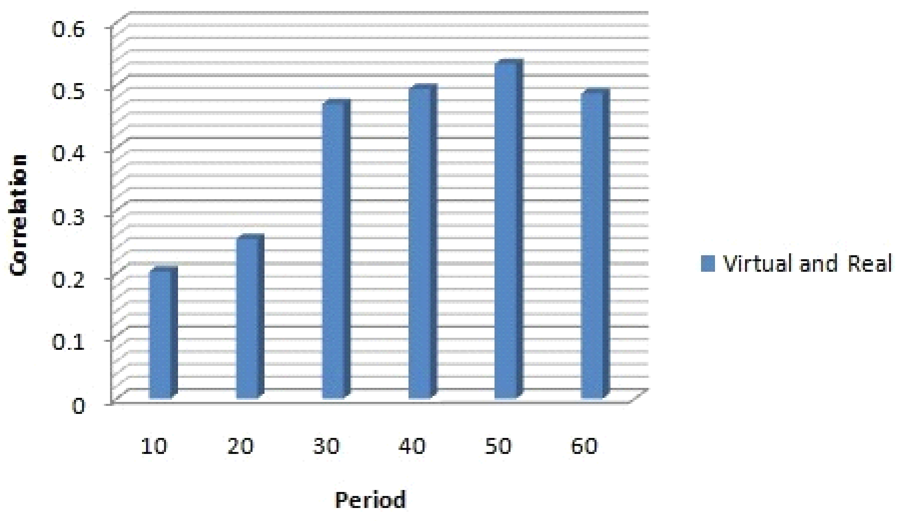


Diagram 3–E correlations of flows

Period	Correlation
10	0. 202945
20	0. 253883
30	0. 469682
40	0. 493338
50	0. 533612

60	0. 486175

Table 3‑C table of flow correlations

From the diagram, we can see that the correlation at the period 10 and period 20 is lower than other four periods. The reason for this is because with the short period time, there is no enough flow information let the system do the prediction base on it. Consider the diagram below. The shorter of the prediction period is, the less of the vehicles will be in it. The prediction can be done only if vehicles are really nearby each other, When the period equal to 10 seconds. So the correlation here is low. At period equal to 20 seconds, the prediction can be done when vehicles a litter bit farther compare to period equal to 10. Also the correlation will be higher. Most of prediction can be done, at period equal to 30 second, so the correlation is the highest in these three. The correlation will keep an increasing trend until period equal to 50 seconds because of that reason. At period equal to 60 second, it goes down again. But the accuracy goes down, although the most of prediction can be done. That is why correlation goes down.

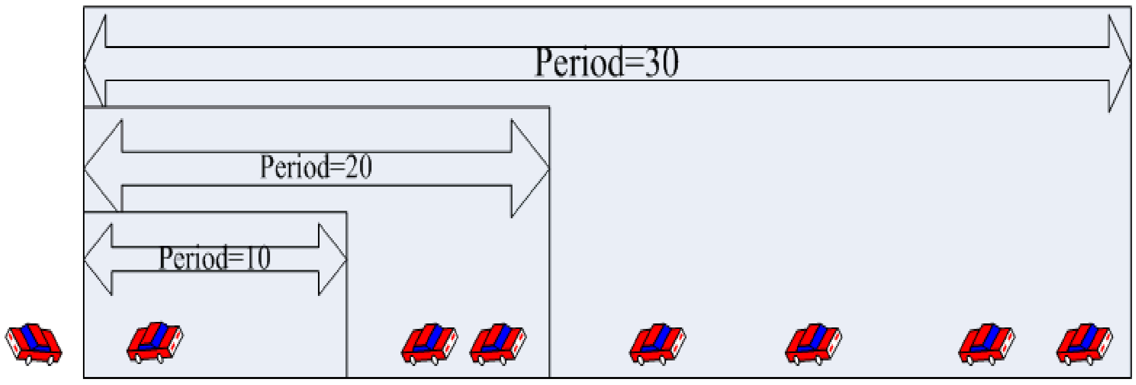


Diagram 3‑F Car streams

3.1.4. Correlation with Different Prediction Distance

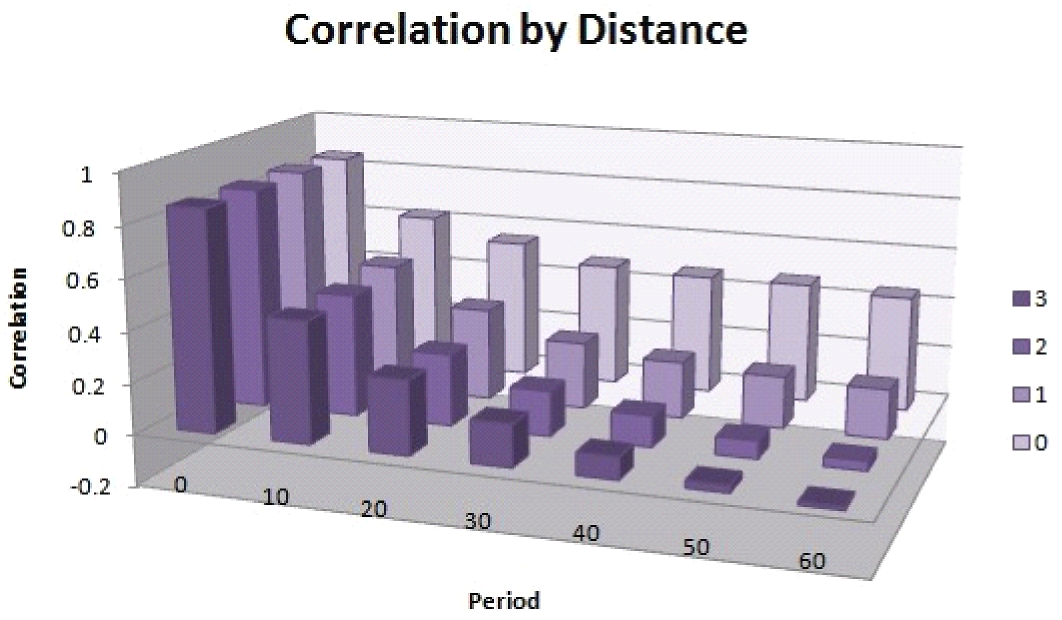


Diagram 3-G Correlations of distances

Period	Distance 0	Distance 1	Distance 2	Distance 3
0	0.853058679	0.860798464	0.859487434	0.860532677
10	0.62759553	0.50262939	0.474740766	0.469889581
20	0.547418107	0.357796857	0.278296414	0.285759368
30	0.480132826	0.259162388	0.174660729	0.163407094
40	0.467392513	0.219327341	0.121433129	0.085715714
50	0.468234721	0.200977221	0.068501757	0.027542073
60	0.44796614	0.190452959	0.036761718	-0.012382269

Table 3-E table of distance correlations

This diagram is the correlation between the prediction number and reality number vehicles in particular VSA with different period and different distance. Distance is how long the VSA region, which had made this prediction, to the predicted VSA region.

From the diagram, with the increasing of the distance, the accuracy of prediction is decreased. The decreasing from distance zero to distance one is because distance one only include for neighboring VSA region, there are also four neighboring VSA region which located at four corner do not include, because the distances between them are square root two which bigger than one. Consider the diagram below, from white region to read regions distance are one, but from white region to the black region is square root two. Some vehicles many flow out or in from the white region to these black regions directly, these number do not add in our correlation, so the distance one correlation lower than distance zero.

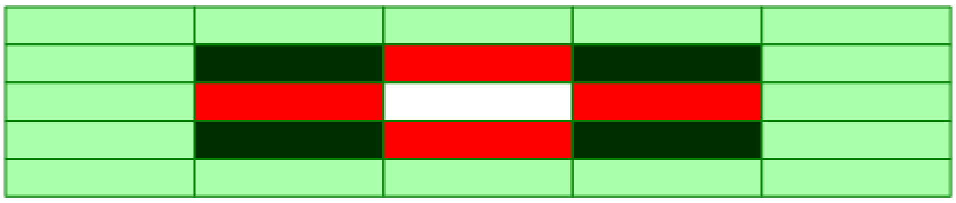


Diagram 3–H Distance of VSA

From distance one to distance two and from distance two to distance three also a fall down. That is because the long distance prediction has been made by virtual flow. The predicting virtual flow which we base on flow cannot achieve 100% accuracy, so these not 100% virtual flows to predict other virtual flow will even lower.

3.2. Correctness

The correlations we calculate are different from the correlations calculated in EXCEL. But the differences are tiny.

We list two groups of different correlations as below:

EXPERIMENT	PERIOD (SEC)	ACCESS	EXCEL	SQUARE DEVIATION
231	10	0.584842908710844	0.585392057	1.50782E-07
231	20	0.512266808664885	0.512717748	1.01673E-07
231	30	0.436409959281383	0.436789776	7.21304E-08
231	40	0.381207616102774	0.381540258	5.53253E-08
231	50	0.370301099166551	0.37062621	5.28485E-08
231	60	0.361884771592126	0.362203894	5.09196E-08

Table 3–F: Correlations between prediction and reality

EXPERIMENT	PERIOD (SEC)	ACCESS	EXCEL	SQUARE DEVIATION
231	10	0.836881721132599	0.837667526	3.08744E-07
231	20	0.823545015266334	0.824269967	2.62777E-07
231	30	0.806282091130769	0.806983816	2.46209E-07
231	40	0.787617571971427	0.788304847	2.36174E-07
231	50	0.780009626879439	0.780694447	2.34489E-07
231	60	0.770429275748161	0.771108667	2.30786E-07

Table 3–G: Correlations between ideal prediction and reality

We find that the correlation results are different between Access database and Excel, the main reason lead to this is because the calculation accuracy used in Access and Excel is different. Access use double float number in calculation, Excel only use float number in calculation. So, the accuracy will be loss during the internal calculation in Excel. We use the correlation formula calculate the relation between the result got from Access and Excel, it is exactly the same.

4. Conclusion

The project is concerned with short term traffic forecast challenges in Vehicular Ad Hoc Network. The traffic forecast is one of the main applications which vehicular network typically support.

In this paper, we implemented an algorithm based on the micro level flow information by concatenating and merging this micro level information to compose the macro level flows under Street Random Waypoint mobility model, which is separated by many pre-defined VSAs. These flows information were recorded as references to guarantee the future traffic prediction.

This thesis achieves the traffic prediction not only for the neighboring VSA regions but also the remote VSA regions in different length of prediction period. We evaluate our project performance, and the prediction system is working as our expectation. According the correlation definition, the relationship between the prediction number of vehicles and the reality number of vehicles in VSA regions is medium degree. This is because two main reasons. Firstly, the message passing in VSAs are “best effort”, which would miss some vehicles when leader calculate its members in VSA. Secondly, our prediction is done by a Random Function, this Random Function can not predict correctly every time.

The future work can be focused on two main disadvantages mentioned above, the VSA “best effort” and current algorithm we developed.

5. Reference

[1] Jagannathan Sarangapani. In Wireless ad hoc and sensor networks: Protocols, performance, and Control ISBN: 0824726758

[2] Chapter 17 vehicular networks and application in the book: Middleware for Network Eccentric and Mobile Applications, Benoît Garbinato, Hugo Miranda and Luís Rodrigues. 2009, Approx. 465 p, Hardcover. ISBN: 978-3-540-89706-4

[3] Shlomi Dolev, Seth Gilbert, Limor Lahiani, Nancy Lynch, Tina Nolte. Virtual Stationary Automata for Mobile Networks. 9th International Conference on Principles of Distributed Systems (OPODIS), December, 2005

[4] Lei Nie and Wei Li. Robust Software Infrastructure for Mobile Ad Hoc Networks, February 7, 2007

[5] http://jist.ece.cornell.edu/

[6] http://www.aqualab.cs.northwestern.edu/projects/STRAW/index.php

[7] Yaqin Wang Yue Chen Minggui Qin Yangyong Zhu. Dynamic Traffic Prediction Based on Traffic Flow Mining. Intelligent Control and Automation, 2006. WCICA 2006.

[8] Shlomi Dolev, Elad Schiller, and Jennifer L. Welch. Random walk for self-stabilizing group communication in ad-hoc network. In SRDS, pages 70-79. IEEE Computer Society, 2002.

[9] S. Dolev, S. Gilbert, L. Lahiani, N. Lynch, and T. Nolte. Virtual stationary automata for mobile networks. Technical Report MIT-LCSTR-979, MIT CSAIL, Cambridge, MA 02139, January 2005.

Appendix A

A.1. Instruction for downloading and installing JiSTSwans and Eclipse

To run simulation with our program you need to have JiST and JDK (NOT JRE) 5.0 or later version installed. You may also want to use Eclipse gives us a much easy way to development and deployment.

- JiST and Swans are available here:
  http://jist.ece.cornell.edu/code/jist-swans-1.0.6-src.tar.gz
- Strand JDK is available here:
  http://java.sun.com/javase/downloads/index_jdk5.jsp
- Eclipse IDE for Java Developers is available here:
  http://www.eclipse.org/downloads/

The first thing you need to do is to install JDK. Follow the installation instruction on the download webpage to install JDK.

1. After installed the JDK, you can extract JiSTSwans and Eclipse to any folder.
2. Choose the menu file, click import.
3. Choose import “exist projects into workspace”.
4. Select the directory which you extracted the JistSwans to.
5. At last, follow the instruction until the whole the whole project is imported to your eclipse’s current workspace.

A.2. Instruction for putting source code into simulator

Our code is compressed in a RAR file. You can extract the file to any folders.

Then, import it just like import JistSwans which we described before.

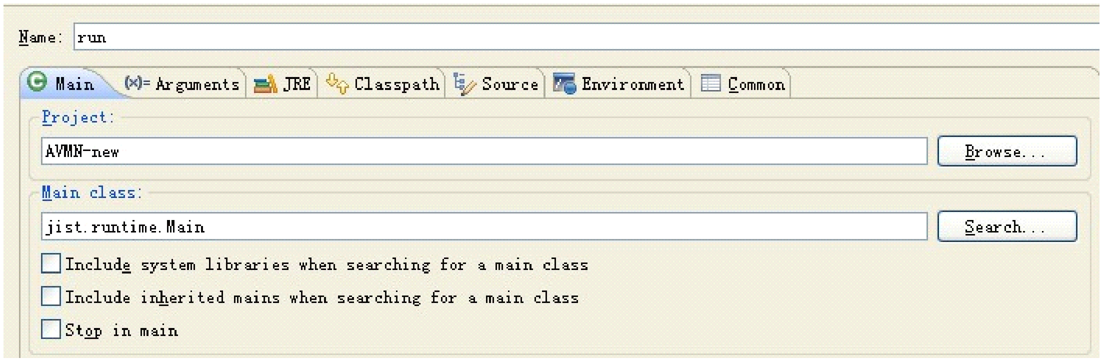
A.3. Instruction for running simulation and parameters setting

The main function is inGenricDriver.java. To run the simulation you need to:

1. Select the Run menu item from the Run menu. It will open a new window named "

"Create, manage and run configurations".

2. Double click "Java Application" on the left side in the window. This operation will create a new configuration. Enter "jist.runtime.Main" in the Main Class and click "Arguments" tag on the top. (Diagram 0–A)
3. In the "arguments" tag, enter: “jist.swans.Main” + “prediction.GenericDriver” + " StreetMobilityDemo.xml” (Diagram 0–B).


Name: run

Project: AVMN-new

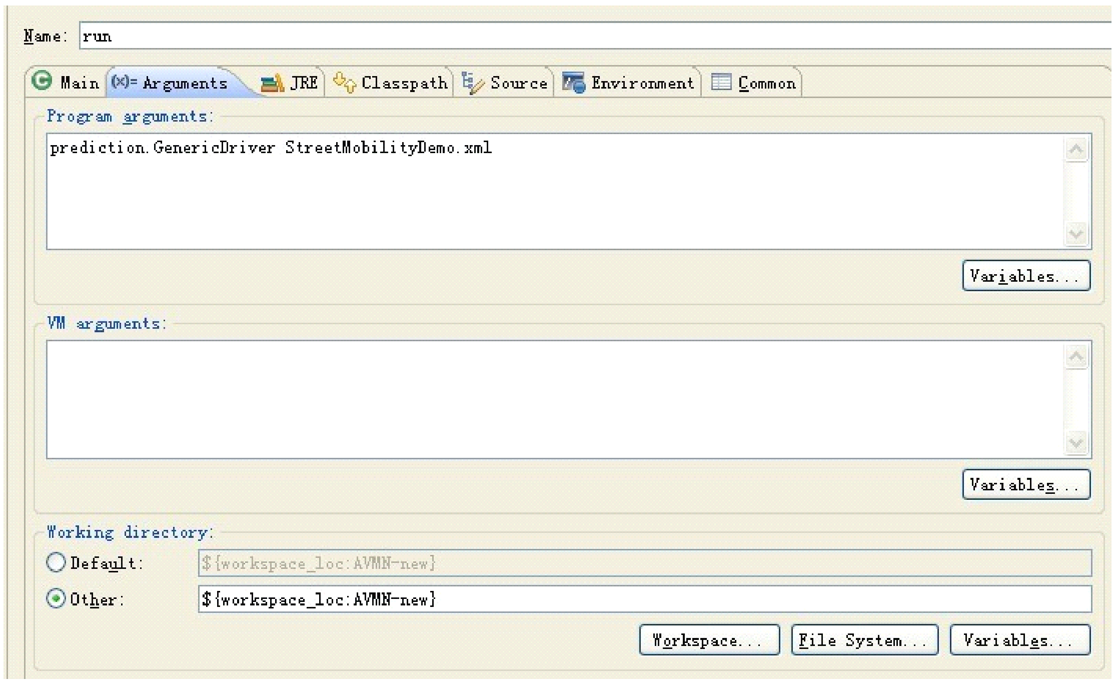
Main class: jist.runtime.Main

☐ Include system libraries when searching for a main class

☐ Include inherited mains when searching for a main class

☐ Stop in main

Diagram 0–A


Name: run

Program arguments: prediction.GenericDriver StreetMobilityDemo.xml

VM arguments:

Working directory:

○ Default: ${workspace_loc:AVMN-new}

◉ Other: ${workspace_loc:AVMN-new}

Diagram 0–B

A.4. XML Configuration

We use XML file (StreetMobilityDemo.xml) instead of command line parameters. The important properties are list here:

1. Duration: The total simulation time
2. Resolution: end before that time
3. start Time: start after that time
4. mobility: mobility model
5. nodes: number of nodes
6. placement: placement mode
7. useVisualizer: whether to display the graphic interface

A.5. Build and execute project remotely

We introduced apache ant (http://ant.apache.org/) to help us build the project without eclipse. So we can access remote server via SSH. This is important, because the project may take hours to get a result. Building and executing the project remotely can help us and successes a lot and release us from the screen of server in the laboratory.

The guild line to build and run our project as below:

1. Put build.xml to the same directory which contains the folder of our project.
2. Rename the project directory to "prediction"
3. Change you current working directory to the directory which contains project and build.xml
4. Execute "ant clean.classes" to clean old java classes
5. Execute "ant" to compile java code
6. Execute ant run to run the project

The detail of build.xml you can find in our source code.

A.6. File list of the source code package

Interface Summary	
GenericDriver.SimulationInterface	Interface for time simulating entries There are: GenericDriver.SimulationInterface.benchmark(), GenericDriver.SimulationInterface.inodePulse(int), GenericDriver.SimulationInterface.sensorUpdate(int), GenericDriver.SimulationInterface.statistics(int),

	GenericDriver.SimulationInterface.vsaGPSUpdate(int), and GenericDriver.SimulationInterface.vsaTimeSlice(int). The default implementation is GenericDriver.Sim.
NodeInterface	Interface for Node in the simulation.
VSAInterface	Interface for VSA in the simulation.

A.7.Table 0–A Interface of prediction project

A.8.

Class Summary	
DoubleLoop	Class for simulating a mobility model (For Debugging) At the developing of our prediction model, we develop a simple mobility model to simulate some idea traffic condition.
Flow	Class for Flow Message The flow message is designed for statistics and prediction.
GenericDriver	Class of simulating main entry The main entry of Traffic Prediction project.
GenericDriver.Sim	this is the implementation of SimulationInterface.
Guardst	Class for joinreq pair and guard pair which consist of variable addr type of Object, and variable time type of long, the pairs can be insert into the corresponding list.
LoopPlacement	Class for placing node to a circle (Designed for debugging) At the developing of our prediction model, we develop a simple mobility model to simulate some idea traffic condition.
MessageUtil	Class for parsing object to message and building message Utilize for parser message-object and generate message-bytes and record text(String) to visualization tool
Node	Class for simulating Node Each Node has four elements: bcNode, iNode, sensor and vsa.
Path	The class for implementing path algorithm Rewrite the hashcod

	e() and equals() Just Compare the from and to region
Pred	Class for recording prediction information to Comma Separated Values (CSV) file The prediction class Made by Leader of VSA Our prediction is based on the volumes of grouped flows and base number.
ProLocation	Class for Pro-Location (location and time tuple) Processor location tuple.
ReceiverModel	Class for message receiving and relay Message receive capability of Processor
Recorder	Class for recording Class Result and Pred information.
Region	Class for a VSA region It contains some property of a VSA region Such as X,Y coordinate and index ...
Result	Class for recording statistic information to Comma Separated Values (CSV) file Made by Leader of VSA Set for recording
SenderModel	Class for initialize nodes which implement the interface InitNodeInterface.
Sensor	Class for simulating AVMN (Autonomous Virtual Mobile Nodes) Class of Sensor for each processor.
StateRefresh	Class for refreshing state It set for each item in AVMN state buffer.
Timer	Class for timer of InitNode, used to delay message sending to avoid message collision.
TimerVsa	Class for timer of VSA, used to delay the process and invoke some certain function of VSA periodically.
VFlow	Class for Virtual Flow Message Note: this class has a natural ordering that is inconsistent with equals.
Vsa	class for simulating VSA (Virtual Stationary Automaton) Class for initialize nodes which implement the interface InitNodeInterface.
VSAFlows	class for prediction and statistic modular.
VSAState	Class for state of VSA This class maintain the information of Vi

	rtual Stationary Automata.

A.9.Table 0–B Class of prediction project

A.10.

Enum Summary	
EnumVSAStatus	Enumeration type for VSA status which includes: starting, trying, guard and leader

A.11. Table 0–C Emulation of prediction project

A.12. Named Queries in Database

Query Name	**Detail**

Fixed_FlowOut	SELECT RealFlow.Experiment, RealFlow.Predicting_Index AS VSA_Index, Sum(RealFlow.Num) AS Num_flowOut, RealFlow.From_past AS Start_Time, ([RealFlow].[To_now]-[RealFlow].[From_past]) AS Period
	FROM RealFlow
	GROUP BY RealFlow.Experiment, RealFlow.Predicting_Index, RealFlow.From_past, ([RealFlow].[To_now]-[RealFlow].[From_past])
	ORDER BY ([RealFlow].[To_now]-[RealFlow].[From_past]);

Fixed_flowIn	SELECT RealFlow.Experiment, RealFlow.Predicted_Index AS VSA_Index, Sum(RealFlow.Num) AS Num_flowIn, RealFlow.From_past AS Start_Time, ([RealFlow].[To_now]-[RealFlow].[From_past]) AS Period
	FROM RealFlow
	GROUP BY RealFlow.Experiment, RealFlow.Predicted_Index, RealFlow.From_past, ([RealFlow].[To_now]-[RealFlow].[From_past])
	ORDER BY ([RealFlow].[To_now]-[RealFlow].[From_past]);

Fixed_flowIn_ flowOut	(SELECT Fixed_flowIn.Experiment AS Experiment, Fixed_flowIn.VSA_Index AS VSA, Fixed_flowIn.Num_flowIn AS Num, Fixed_flowIn.Start_Time AS Start_Time, Fixed_flowIn.Period AS Period FROM Fixed_flowIn) UNION (SELECT Fixed_flowOut.Experiment AS Experiment, Fixed_flowOut.VSA_Index AS VSA, -(Fixed_flowOut.Num_flowOut) AS Num, Fixed_flowOut.Start_Time AS Start_Time, Fixed_flowOut.Period AS Period FROM Fixed_flowOut);
Fixed_num	(SELECT Result.Experiment, Result.VSA_Index, Result.Time, Fixed_flowIn_flowOut.Period, IIF((AVG(Result.Num)+SUM(Fixed_flowIn_flowOut.Num))>0,(AVG(Result.Num)+SUM(Fixed_flowIn_flowOut.Num)),0) AS Num FROM Result INNER JOIN Fixed_flowIn_flowOut ON (Result.Experiment = Fixed_flowIn_flowOut.Experiment) AND (Result.VSA_Index = Fixed_flowIn_flowOut.VSA) AND (Result.Time = Fixed_flowIn_flowOut.Start_Time+Fixed_flowIn_flowOut.Period) GROUP BY Result.Experiment, Result.VSA_Index, Result.Time, Fixed_flowIn_flowOut.Period) UNION (SELECT Result.Experiment, Result.VSA_Index, Result.Time, 0, Result.Num FROM Result);
Pred	SELECT Prediction.Experiment, Prediction.Predicted_Index, Prediction.Furture, Sum(Prediction.Num) AS Num, Prediction.Furture-Prediction.Now AS Period FROM Prediction GROUP BY Prediction.Experiment, Prediction.Predicted_Index, Prediction.Furture, (Prediction.Furture-Prediction.Now);
all_num	SELECT Result.Experiment, Result.VSA_Index, Result.Time, Fixed_num.Period, Result.Num AS resN, Fixed_num.Num AS fixN, Pred.Num AS preN

	FROM (Pred INNER JOIN Fixed_num ON (Pred.Experiment = Fixed_num.Experiment) AND (Pred.Predicted_Index = Fixed_num.VSA_Index) AND (Pred.Furture = Fixed_num.Time) AND (Pred.Period = Fixed_num.Period)) INNER JOIN Result ON (Fixed_num.Experiment = Result.Experiment) AND (Fixed_num.VSA_Index = Result.VSA_Index) AND (Fixed_num.Time = Result.Time)
	WHERE (((Pred.Experiment)=[Fixed_num].[Experiment]));

perAvg	SELECT all_num.Experiment, Avg([all_num].[preN]*[all_num].[resN]) AS AvgOfPreReal, Avg(all_num.[preN]) AS AvgOfPre, Avg(all_num.[resN]) AS AvgOfReal, StDev(all_num.[preN]) AS StdOfPre, StDev(all_num.[resN]) AS StdOfReal, all_num.Period
	FROM all_num
	GROUP BY all_num.Experiment, all_num.Period;

perCor	SELECT perAvg.Experiment, perAvg.Period, ([perAvg].[AvgOfPreReal]-([perAvg].[AvgOfPre]*[perAvg].[AvgOfReal]))/([perAvg].[StdOfReal]*[perAvg].[StdOfPre]) AS Cor
	FROM perAvg
	GROUP BY perAvg.Experiment, perAvg.Period, ([perAvg].[AvgOfPreReal]-([perAvg].[AvgOfPre]*[perAvg].[AvgOfReal]))/([perAvg].[StdOfReal]*[perAvg].[StdOfPre]);

fixAvg	SELECT all_num.Experiment, all_num.Period, Avg([all_num].[fixN]*[all_num].[resN]) AS AvgOfFixReal, Avg(all_num.fixN) AS AvgOfFix, Avg(all_num.resN) AS AvgOfReal, StDev(all_num.fixN) AS StdOfFix, StDev(all_num.resN) AS StdOfReal
	FROM all_num
	GROUP BY all_num.Experiment, all_num.Period;

fixCor	SELECT fixAvg.Experiment, fixAvg.Period, ([fixAvg].[AvgOfFixReal]-([fixAvg].[AvgOfFix]*[fixAvg].[AvgOfReal]))/([fixAvg].[StdOfReal]*[fixAvg].[StdOfFix]) AS Cor
	FROM fixAvg

	GROUP BY fixAvg.Experiment, fixAvg.Period, ([fixAvg].[AvgOfFixReal]-([fixAvg].[AvgOfFix]*[fixAvg].[AvgOfReal]))/([fixAvg].[StdOfReal]*[fixAvg].[StdOfFix]);

dis01	SELECT p.Experiment, Result.VSA_Index, Result.Time, Result.Num AS Real_Num, IIf(Sum([p].[Num])>0,Sum([p].[Num]),0) AS Predicted_Num, [p].[Furture]-[p].[Now] AS Period, MAX(p.distance) AS distance
	FROM Result, (SELECT Prediction.Experiment, Prediction.Num, Prediction.Predicted_Index, Prediction.furture, Prediction.Now, Sqr(([a].[x]-[b].[x])*([a].[x]-[b].[x])+([a].[y]-[b].[y])*([a].[y]-[b].[y])) as distance
	FROM Prediction, coordinate AS a, coordinate AS b
	WHERE (Sqr(([a].[x]-[b].[x])*([a].[x]-[b].[x])+([a].[y]-[b].[y])*([a].[y]-[b].[y]))<=1) AND (Prediction.Predicted_Index = a.VSA_Index) AND (Prediction.Predicting_Index = b.VSA_Index)
	) AS p
	WHERE (((p.Furture)=[Result].[Time]) AND ((p.Predicted_Index)=[Result].[VSA_Index]) AND ((p.Experiment)=[Result].[Experiment]))
	GROUP BY p.Experiment, Result.VSA_Index, Result.Time, Result.Num, [p].[Furture]-[p].[Now]
	ORDER BY [p].[Furture]-[p].[Now];

dis012	SELECT p.Experiment, Result.VSA_Index, Result.Time, Result.Num AS Real_Num, IIf(Sum([p].[Num])>0,Sum([p].[Num]),0) AS Predicted_Num, [p].[Furture]-[p].[Now] AS Period, Max(p.distance) AS distance
	FROM Result, (SELECT Prediction.Experiment, Prediction.Num, Prediction.Predicted_Index, Prediction.furture, Prediction.Now, Sqr(([a].[x]-[b].[x])*([a].[x]-[b].[x])+([a].[y]-[b].[y])*([a].[y]-[b].[y])) as distance
	FROM Prediction, coordinate AS a, coordinate AS b
	WHERE (Sqr(([a].[x]-[b].[x])*([a].[x]-[b].[x])+([a].[y]-[b].[y])*([a].[y]-[b].[y]))<=2 AND Sqr(([a].[x]-[b].[x])*([a].[x]-[b].[x])+([a].[y]-[b].[y])*([a].[y]-[b].[y]))>1 OR Sqr(([a].[x]-[b].[x])*([a].[x]-[b].[x])+([a].[y]-[b].[y])*([a].[y]-[b].[y])) =0) AND (Prediction.Predicted_Index = a.VSA_Index) AND (Prediction.Predi

	cting_Index = b.VSA_Index)
	
	) AS p
	
	WHERE (((p.Furture)=[Result].[Time]) AND ((p.Predicted_Index)=[Result].[VSA_Index]) AND ((p.Experiment)=[Result].[Experiment]))
	
	GROUP BY p.Experiment, Result.VSA_Index, Result.Time, Result.Num, [p].[Furture]-[p].[Now]
	
	ORDER BY [p].[Furture]-[p].[Now];

dis0123	SELECT p.Experiment, Result.VSA_Index, Result.Time, Result.Num AS Real_Num, IIf(Sum([p].[Num])>0,Sum([p].[Num]),0) AS Predicted_Num, [p].[Furture]-[p].[Now] AS Period, Max(p.distance) AS distance
	
	FROM Result, (SELECT Prediction.Experiment, Prediction.Num, Prediction.Predicted_Index, Prediction.furture, Prediction.Now, Sqr(([a].[x]-[b].[x])*([a].[x]-[b].[x])+([a].[y]-[b].[y])*([a].[y]-[b].[y])) as distance
	
	FROM Prediction, coordinate AS a, coordinate AS b
	
	WHERE (Sqr(([a].[x]-[b].[x])*([a].[x]-[b].[x])+([a].[y]-[b].[y])*([a].[y]-[b].[y]))<=3 AND Sqr(([a].[x]-[b].[x])*([a].[x]-[b].[x])+([a].[y]-[b].[y])*([a].[y]-[b].[y]))>2 OR Sqr(([a].[x]-[b].[x])*([a].[x]-[b].[x])+([a].[y]-[b].[y])*([a].[y]-[b].[y]))=0) AND (Prediction.Predicted_Index = a.VSA_Index) AND (Prediction.Predicting_Index = b.VSA_Index)
	
	) AS p
	
	WHERE (((p.Furture)=[Result].[Time]) AND ((p.Predicted_Index)=[Result].[VSA_Index]) AND ((p.Experiment)=[Result].[Experiment]))
	
	GROUP BY p.Experiment, Result.VSA_Index, Result.Time, Result.Num, [p].[Furture]-[p].[Now]
	
	ORDER BY [p].[Furture]-[p].[Now];

dis01234	SELECT p.Experiment, Result.VSA_Index, Result.Time, Result.Num AS Real_Num, IIf(Sum([p].[Num])>0,Sum([p].[Num]),0) AS Predicted_Num, [p].[Furture]-[p].[Now] AS Period, MAX(p.distance) AS distance
	
	FROM Result, (SELECT Prediction.Experiment, Prediction.Num, Prediction.Predicted_Index, Prediction.furture, Prediction.Now, Sqr(([a].[x]-[b].[x])*([a].[x]-[b].[x])+([a].[y]-[b].[y])*([a].[y]-[b].[y])

	) as distance
	FROM Prediction, coordinate AS a, coordinate AS b
	WHERE (Sqr(([a].[x]-[b].[x])*([a].[x]-[b].[x])+([a].[y]-[b].[y])*([a].[y]-[b].[y]))>3 AND Sqr(([a].[x]-[b].[x])*([a].[x]-[b].[x])+([a].[y]-[b].[y])*([a].[y]-[b].[y]))<=4 OR Sqr(([a].[x]-[b].[x])*([a].[x]-[b].[x])+([a].[y]-[b].[y])*([a].[y]-[b].[y]))=0) AND (Prediction.Predicted_Index = a.VSA_Index) AND (Prediction.Predicting_Index = b.VSA_Index)
	) AS p
	WHERE (((p.Furture)=[Result].[Time]) AND ((p.Predicted_Index)=[Result].[VSA_Index]) AND ((p.Experiment)=[Result].[Experiment]))
	GROUP BY p.Experiment, Result.VSA_Index, Result.Time, Result.Num, [p].[Furture]-[p].[Now]
	ORDER BY [p].[Furture]-[p].[Now];

dis01AVG	SELECT dis01.Experiment, Avg([dis01].[Real_Num]*[dis01].[Predicted_Num]) AS AvgOfPreReal, Avg(dis01.[Real_Num]) AS AvgOfPre, Avg(dis01.[Predicted_Num]) AS AvgOfReal, StDev(dis01.[Predicted_Num]) AS StdOfPre, StDev(dis01.[Real_Num]) AS StdOfReal, dis01.Period
	FROM dis01
	GROUP BY dis01.Experiment, dis01.Period;

dis012AVG	SELECT dis012.Experiment, Avg([dis012].[Real_Num]*[dis012].[Predicted_Num]) AS AvgOfPreReal, Avg(dis012.[Real_Num]) AS AvgOfPre, Avg(dis012.[Predicted_Num]) AS AvgOfReal, StDev(dis012.[Predicted_Num]) AS StdOfPre, StDev(dis012.[Real_Num]) AS StdOfReal, dis012.Period
	FROM dis012
	GROUP BY dis012.Experiment, dis012.Period;

dis0123AVG	SELECT dis0123.Experiment, Avg([dis0123].[Real_Num]*[dis0123].[Predicted_Num]) AS AvgOfPreReal, Avg([Real_Num]) AS AvgOfPre, Avg([Predicted_Num]) AS AvgOfReal, StDev([Predicted_Num]) AS StdOfPre, StDev([Real_Num]) AS StdOfReal, dis0123.Period

	FROM dis0123 GROUP BY dis0123.Experiment, dis0123.Period;
dis01234AVG	SELECT dis01234.Experiment, Avg([dis01234].[Real_Num]*[dis01234].[Predicted_Num]) AS AvgOfPreReal, Avg(dis01234.Real_Num) AS AvgOfPre, Avg(dis01234.Predicted_Num) AS AvgOfReal, StDev(dis01234.Predicted_Num) AS StdOfPre, StDev(dis01234.Real_Num) AS StdOfReal, dis01234.Period FROM dis01234 GROUP BY dis01234.Experiment, dis01234.Period;
dis01Cor	SELECT dis01Avg.Experiment, dis01Avg.Period, ([dis01Avg].[AvgOfPreReal]-([dis01Avg].[AvgOfPre]*[dis01Avg].[AvgOfReal]))/([dis01Avg].[StdOfReal]*[dis01Avg].[StdOfPre]) AS Cor FROM dis01Avg GROUP BY dis01Avg.Experiment, dis01Avg.Period, ([dis01Avg].[AvgOfPreReal]-([dis01Avg].[AvgOfPre]*[dis01Avg].[AvgOfReal]))/([dis01Avg].[StdOfReal]*[dis01Avg].[StdOfPre]);
dis012Cor	SELECT dis012Avg.Experiment, dis012Avg.Period, ([dis012Avg].[AvgOfPreReal]-([dis012Avg].[AvgOfPre]*[dis012Avg].[AvgOfReal]))/([dis012Avg].[StdOfReal]*[dis012Avg].[StdOfPre]) AS Cor FROM dis012Avg GROUP BY dis012Avg.Experiment, dis012Avg.Period, ([dis012Avg].[AvgOfPreReal]-([dis012Avg].[AvgOfPre]*[dis012Avg].[AvgOfReal]))/([dis012Avg].[StdOfReal]*[dis012Avg].[StdOfPre]);
dis0123Cor	SELECT dis0123AVG.Experiment, dis0123AVG.Period, ([dis0123AVG].[AvgOfPreReal]-([dis0123AVG].[AvgOfPre]*[dis0123AVG].[AvgOfReal]))/([dis0123AVG].[StdOfReal]*[dis0123AVG].[StdOfPre]) AS Cor FROM dis0123AVG GROUP BY dis0123AVG.Experiment, dis0123AVG.Period, ([dis0123AVG].[AvgOfPreReal]-([dis0123AVG].[AvgOfPre]*[dis0123AVG].[AvgOfReal]))/([dis012

	3AVG].[StdOfReal]*[dis0123AVG].[StdOfPre]);

dis01234Cor	SELECT dis01234AVG.Experiment, dis01234AVG.Period, ([dis01234AVG].[AvgOfPreReal]-([dis01234AVG].[AvgOfPre]*[dis01234AVG].[AvgOfReal]))/([dis01234AVG].[StdOfReal]*[dis01234AVG].[StdOfPre]) AS Cor
	FROM dis01234AVG
	WHERE dis01234AVG.STDofPre <> 0
	GROUP BY dis01234AVG.Experiment, dis01234AVG.Period, ([dis01234AVG].[AvgOfPreReal]-([dis01234AVG].[AvgOfPre]*[dis01234AVG].[AvgOfReal]))/([dis01234AVG].[StdOfReal]*[dis01234AVG].[StdOfPre]);

disCorComp	SELECT dis0123Cor.Experiment, dis0123Cor.Period, dis01Cor.Cor AS Cor01, dis012Cor.Cor AS Cor02, dis0123Cor.Cor AS Cor03, dis01234Cor.Cor AS Cor04
	FROM ((dis0123Cor INNER JOIN dis012Cor ON (dis0123Cor.Period = dis012Cor.Period) AND (dis0123Cor.Experiment = dis012Cor.Experiment)) INNER JOIN dis01Cor ON (dis012Cor.Period = dis01Cor.Period) AND (dis012Cor.Experiment = dis01Cor.Experiment)) INNER JOIN dis01234Cor ON (dis01Cor.Period = dis01234Cor.Period) AND (dis01Cor.Experiment = dis01234Cor.Experiment);

A.13. Changes in the simulator

Add two variables in Constants.java. It is located in package jist.swans.

Added variables are:

1. public static final int PLACEMENT_CRICLE = 4;
2. public static final int MOBILITY_DoubleLoop = 13;

They are used to handle one new defined mobility model—Double Loop. And one new defined placement model circle placements.

Both of they are used in our defined mobility model for debugging.

A.14. Changes in the VSA deployment

1. For updating the performance of the project, we used Message interface instead of

parsing Message to String by ourselves. New message will be introduced to the system should implements Message Interface. It provide by Swans in the package "jist.swans.misc". (Note: Remember implement java.lang.cloneable interface too)

2. For structure the code more clearly, we decouple the internal class: Vsa, senderModel, reciverModel and Sensor to four standalone classes. So, the code for class Node reduced from more than 5000 lines to less than 500 lines. Also, we introduce sub-package message, misc, mode, sensor, statistic and vsa in the avmn package.
3. For speedup our work plan, we used some interfaces and functions providing by Java standard library such as contains, sort and cloneable etc. So, we do not need to maintain some existing algorithm by ourselves, such as sorting algorithm, containing, cloning and so on.
4. Because Jist only works under JDK 1.4 currently. It does not support some functions like generic and new for loop. But we can use these functions providing by JDK 5 or later by implements DoNotRewrite interface in the class.

A.15. Simulation Environment

Linux black 2.6.28-11-generic #42-Ubuntu SMP x86_64

Dual Core AMD Opteron(tm) Processor 265 * 2

cpu MHz: 1808.251

cache size: 1024 KB

Memory: 8 GB

A.16. Problems unsolved

The simulator throws Exception in more density situation. (When nodes more than 200 nodes)

This is due to the default implementation of MessageQueues in SWANS. For whatever reason, the queue throws an exception instead of simply dropping the message. A workaround is to place a return statement at the line throwing the exception.

Drop message can keep the simulation going, but it will affect the result of correlation.