Summary

SUMO, as a traffic simulation software, can generate realistic traffic trajectory of vehicle by combining with the real road network provided by OpenStreetMap official website. OPNET, a popular network simulation software, establishes network equipment, communication link and protocol model and simulating traffic transmission to evaluates network performance and optimizes its design by, which has been applied to vehicle network communication simulation well.

However, the trajectory files generated by SUMO do not directly support OPNET simulation. Therefore, this project uses Java language to develop a tool that can convert trajectory generated by SUMO into OPNET simulation trajectory, aiming to realize the joint simulation of them, and provide a more real simulation scene for the research of VANET.

Deployment environment

Win 7 + java-1.8.0-openjdk-devel.x86_64

SUMO-1.6.0

OpenStreetMap

OPNET 14.5

Deployment steps

The overall deployment process is shown in Fig.1, and the detailed steps are as follows:

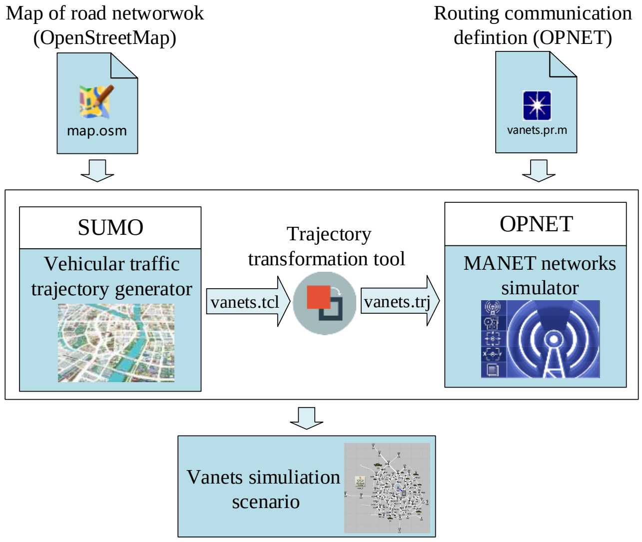

Fig.1 The VANETs simulation based on co-simulation of SUMO and OPNET

Step 1: OSM real map file acquisition (shown as Fig 2.)

Open the OpenStreetMap website (*https://www.openstreetmap.org*) , click "*export*" in the upper right corner, input the longitude and latitude range of the area to be selected in the four blank boxes, and then click "*export*" in the blue background box to generate the OSM map file *map.osm.*

Fig.2.OSM real map file acquisition

Step 2: Generate the ***map.net.xml*** and ***map.poly.xml*** files (shown as Fig.3.)

Put the files ***map.osm*** and ***typemap.xml*** into the new folder named ***map***, and then open the ***map*** folder in cmd command box to run the following two commands in turn:

(1) netconvert --osm-files map.osm -o map.net.xml

(2)polyconvert--net-file map.net.xml--osm-files map.osm --type-file typemap.xml -o map.poly.xml

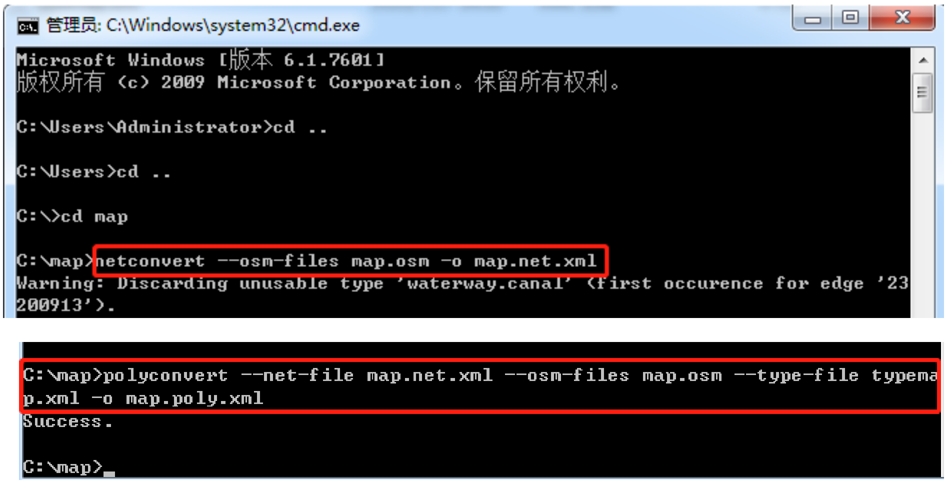

Fig.3. Generate the ***map.net.xml*** and ***map.poly.xml*** files

Step3: Generate ***map.rou.xml*** file (shown as Fig 4.)

Open the ***map*** folder in cmd command box to run the following two commands in turn:

(1) python C:/sumo/tools/randomTrips.py -n map.net.xml -e 100 –l

(2) python C:/sumo/tools/randomTrips.py -n map.net.xml -r map.rou.xml -e 400 -l

(400 indicates the number of vehicles set in the scene, which can be set by the user in combination with the simulation scenario)

```
C:\map>python C:/sumo/tools/randomTrips.py  -n map.net.xml -e 100 -l

C:\map>python C:/sumo/tools/randomTrips.py -n map.net.xml -r map.rou.xml -e 400
-l
calling  C:\Sumo\bin\duarouter -n map.net.xml -r trips.trips.xml --ignore-errors
 --begin 0 --end 400.0 --no-step-log --no-warnings -o map.rou.xml
Success.

C:\map>
```

Fig.4. Generate map.rou.xml file

Step4: Edit configuration file *myConfig.sumocfg.* (shown as Fig 5.)

You can search the file *test.sumocfg* in the *sumo* folder and then change the input codes "<input> ……</input> " to get the file and save it into the *map* folder named as *myConfig.sumocfg*.

```
<?xml version="1.0" encoding="UTF-8"?>

<!-- generated on Tue Apr 28 00:06:20 2020 by Eclipse SUMO Version 1.6.0
This data file and the accompanying materials
are made available under the terms of the Eclipse Public License v2.0
which accompanies this distribution, and is available at
http://www.eclipse.org/legal/epl-v20.html
SPDX-License-Identifier: EPL-2.0
-->

<configuration xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:noNamespaceSchemaLocation="http://sumo.dlr.de

    <input>
        <net-file value="map.net.xml"/>
        <route-files value="map.rou.xml"/>
        <additional-files value="map.poly.xml"/>
    </input>
    <time>
        <begin value="0"/>
        <end value="10000"/>
    </time>

</configuration>
```

Fig.5. Edit configuration file *myConfig.sumocfg*

Step5: Generate SUMO trace file ***ns2 mobility.tcl***.(shown as Fig 6.)

Open the ***map*** folder in cmd command box to run the following two commands in turn:

(1) sumo -c myConfig.sumocfg --fcd-output sumoTrace.xml

(2) traceExporter.py --fcd-input sumoTrace.xml --ns2mobility-output ns2mobility.tcl

```
C:\map>sumo -c myConfig.sumocfg --fcd-output sumoTrace.xml
Loading configuration ... done.
Step #0.00 (7ms ~= 142.86*RT, ~142.86UPS, vehicles TOT 1 ACT 1 BUF 0)
Step #100.00 (3ms ~= 333.33*RT, ~30666.67UPS, vehicles TOT 94 ACT 92 BUF 0)
Step #200.00 (4ms ~= 250.00*RT, ~36500.00UPS, vehicles TOT 189 ACT 146 BUF 0)
Step #300.00 (6ms ~= 166.67*RT, ~30333.33UPS, vehicles TOT 280 ACT 182 BUF 0)
Step #400.00 (4ms ~= 250.00*RT, ~56250.00UPS, vehicles TOT 372 ACT 225 BUF 0)

C:\map>traceExporter.py --fcd-input sumoTrace.xml --ns2mobility-output ns2mobili
ty.tcl
```

Fig.6.Generate SUMO trace file *ns2 mobility.tcl*

Step6: SUMO trace file ***ns2Mobility.tcl*** converts to the file format (****.trj***) supported by OPNET simulation.

(1) Create a new project named ***SUMO_Pretreatment*** in Java IDE and run code ***SUMO_Pretreatment.java***.The result of SUMO trajectory preprocessing will be written to subfolder ***SUMO_F*** by individual vehicle classification.

(2) Create a new project named ***SUMO_To_OPNET_Trj*** in the Java IDE and run the code ***SUMO_To_OPNET_Trj.java.*** The result set of SUMO trajectory preprocessing will be converted into OPNET trajectory file one by one and stored in

the subfolders ***OPNET_Trj.***

Step7: Import OPNET trajectory files into VANETs simulation scenario

Copy the vehicle trajectory files generated in step 6 to the OPNET installation folder ***op_models*** and create a new OPNET project, add nodes and associate trajectories.

One of the advantages of OPNET is that it provides convenient graphical modeling, but sometimes graphical modeling methods do not meet our needs, such as the need to deploy thousands of nodes. It is too cumbersome to deploy these nodes manually one by one, while randomized deployment with a program may result in randomized node locations where a part of the node cluster is clustered together and in some places cannot be deployed to. Therefore, when we randomly deploy nodes, we can randomly generate the coordinates of the next network node; at the same time, in the process of generating node coordinates, we should avoid making the new node coordinates too close to the existing node coordinates.

In response to this requirement, we adopt a text-based model access (EMA), which can precisely define node attributes and can use circular statements to characterize nodes with multiple specific specifications, as follows:

(1) Make a simple network model with only one mobile node under the graphical interface, shown as Fig.7(a).

(2) Select the menu **Topology-->Export Topology-->To EMA** to generate the EMA file (****. em. c***), and find the node whose name attribute is "0" in the EMA file. The code is shown as Fig.7(b)

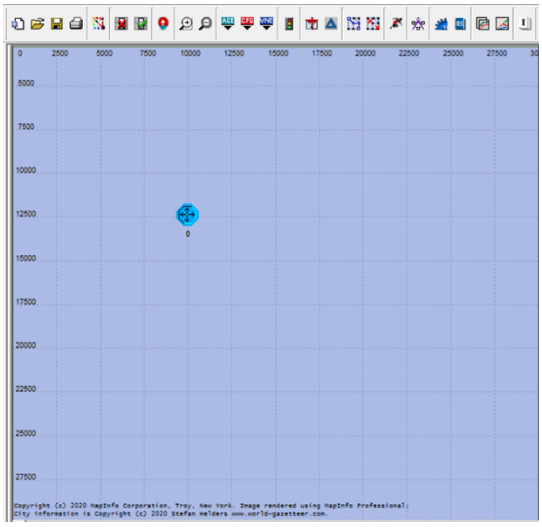

Fig.7(a). A simple network model.

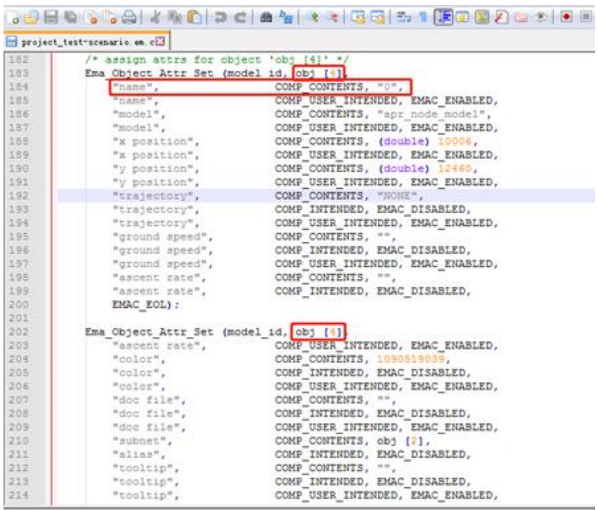

Fig.7(b). EMA file

（3）Add header file as follows Fig.8

```
#include <opnet.h>
#include <ema.h>
#include <opnet_emadefs.h>
#include <opnet_constants.h>
# include <stdlib.h>


/* array for all textlist attributes in model */
Prg_List*       prg_lptr [6];

/* array for all objects in model */
EmaT_Object_Id      obj [13];

# define NODE_NUMBER 800                /*节点数量*/

# define X_MIN 9467.75                  /*场景大小，可以采用SUMO.java输出区域的范围*/
# define X_MAX 13216.94
# define Y_MIN 12293.16
# define Y_MAX 16287.15
# define X_LENGTH 3749.19
# define Y_LENGTH 3993.99

# define MIN_DISTANCE_2 400             /*节点最小距离的平方，这个值越大，节点生成越均匀，这个值理论最大为25 x 25 x 1.414，值越大生成的时间越长*/

int PosChooseAgain;                     /*是否重新生成*/
int NodeNumber;                         /*节点数*/
int i, k;                               /*循环变量*/
char node_name[5];                      /*节点名称*/
char tra[20];                           /*轨迹名称*/
double x_pos,y_pos;                     /*节点坐标*/
double NodeXList[800] , NodeYList[800]; /*节点坐标数组·*/
EmaT_Object_Id wsn_node_objid;          /*对象变量*/
```

Fig.8. Add header file

（4）Next set the properties of the node (show as Fig.9)： we place the code of obj [4] (***ema_obj_attr_set()***) in a for-loop statement, replacing the node name attribute and coordinates (x_postion and y_position) as variables, which makes the other nodes are generated by looping.

```
NodeNumber=0;
for (k=1; k <= NODE_NUMBER; k++){
sprintf(node_name, "%d", k);
sprintf(tra, "10ms_2_Real_SUMO_%d", k);
PosChooseAgain = 1;
while(PosChooseAgain == 1)
{
   x_pos= ((double)rand()/((double)(RAND_MAX) + (double)(1)))*(double)X_LENGTH + X_MIN;
   y_pos= ((double)rand()/((double)(RAND_MAX) + (double)(0)))*(double)Y_LENGTH + Y_MIN;
  for (i=0; i< NodeNumber; i++)
  {
    if (((NodeXList[i]- x_pos)*(NodeXList[i]- x_pos)+(NodeYList[i]- y_pos)*(NodeYList[i]- y_pos))< MIN_DISTANCE_2)
    {
       break;
    }
 }

    if(i == NodeNumber) {
      PosChooseAgain = 0;
    }
}

NodeNumber++;
NodeXList[NodeNumber-1] = x_pos;
NodeYList[NodeNumber-1] = y_pos;
wsn_node_objid= Ema_Object_Create(model_id, OBJ_NT_NODE_MOBILE);

/* assign attrs for object 'obj [4]' */
Ema_Object_Attr_Set (model_id, wsn_node_objid
    "name",                 COMP_CONTENTS, name,
    "name",                 COMP_USER_INTENDED, EMAC_ENABLED,
    "model",                COMP_CONTENTS, "apr_node_model",
    "model",                COMP_USER_INTENDED, EMAC_ENABLED,
    "x position",           COMP_CONTENTS, (double) x_pos,
    "x position",           COMP_USER_INTENDED, EMAC_ENABLED,
    "y position",           COMP_CONTENTS, (double) y_pos,
    "y position",           COMP_USER_INTENDED, EMAC_ENABLED,
    "trajectory",           COMP_CONTENTS, tra,
    "trajectory",           COMP_INTENDED, EMAC_DISABLED,
    "trajectory",           COMP_USER_INTENDED, EMAC_ENABLED,
    "ground speed",         COMP_CONTENTS, "",
    "ground speed",         COMP_INTENDED, EMAC_DISABLED,
    "ground speed",         COMP_USER_INTENDED, EMAC_ENABLED,
    "ascent rate",          COMP_CONTENTS, "",
    "ascent rate",          COMP_INTENDED, EMAC_DISABLED,
    EMAC_EOL);

Ema_Object_Attr_Set (model_id, wsn_node_objid
    "ascent rate",          COMP_USER_INTENDED, EMAC_ENABLED,
    "color",                COMP_CONTENTS, 1090519039,
    "color",                COMP_INTENDED, EMAC_DISABLED,
    "color",                COMP_USER_INTENDED, EMAC_ENABLED,
    "doc file",             COMP_CONTENTS, "",
    "doc file",             COMP_INTENDED, EMAC_DISABLED,
    "doc file",             COMP_USER_INTENDED, EMAC_ENABLED,
```

Fig.9.Set the properties of the node

(5) Open the OPNET console and enter the directory where the modified file ***. em.c** is located and execute the following commands in turn (shown as Fig.10)：

① Enter the command ***op_mkema - m file name*** < without suffix >, after successful execution generates ****.dev32.i0.em.x.***

② Executing the newly created executable file < ***file name. i0. em. x***> to generates a new model file (****. m***)

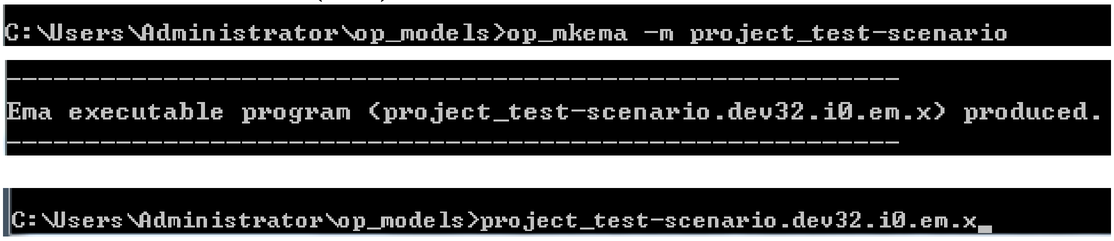

Fig.10. Generate the file (****.m*** **)**

(6) Click the **File->Model Files->Refresh Model Directories** option in the OPNET menu to refresh the model directory, otherwise the newly created model file (**. m*) is not visible in the OPNET. The generated network model, which needs to be imported into the project scenario. And then select **Scenarios->Scenario Components->import** in the project editor, the resulting model is shown in the following figure, and it can be found that the nodes are roughly uniformly distributed and associated with the real road trajectory we generated using SUMO (shown as Fig.11).

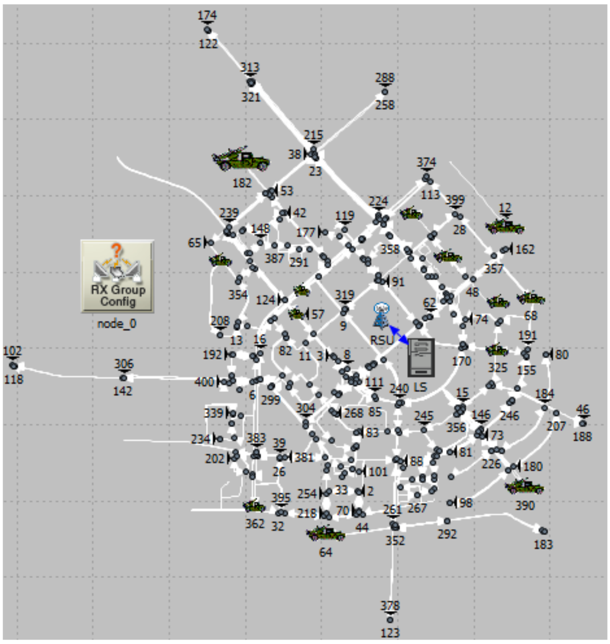

Fig 11. VANETs simulation under real-road network