CHAPTER 4 RESEARCH TOOLS

4.1 Tools Introduction

Lots of tools are required during implementation of proposed research work like SUMO, NS2, Python Perl, MOVE and GCC.

4.1.1 SUMO

SUMO is used for simulation of different traffic model with vehicles, public transport and pedestrians. It is licensed under GNU General Public License. It is open source software. (Spaho et al., 2011)

SUMO was developed by German Aerospace Center in 2001. SUMO functionality includes routing with various sources, different format of maps support etc. SUMO is used in various research areas like route selection and traffic light algorithm or simulating vehicular communication. SUMO framework is used in various projects to extend its functionality.

SUMO traffic simulation requires various own format to represent road networks, vehicles and traffic pattern which could be generated by its own way or by other format. NETGEN is used to generate SUMO road networks or also generated from digital copy from map database. NETCONVERT is used to read network data from other simulators like VISUM, Vissim, or MATsim and also reads other formats like shape files, RoboCup, openDRIVE or Open Street Map. TIGER network is not supported in SUMO. But can be used after converting it into shape files. (Behrisch et al., 2011)
Maps are visible as per Fig. 4.1 on openstreetmap website. It could be converted into SUMO net file as shown in Fig. 4.2 after conversion. Each vehicle is uniquely identified by its name, the departure time, and travelling route through the network. Various other parameters are also available for each vehicle if wants to use. Position, speed, lane properties for departure and arrival are also possible to define. Physical properties of vehicles are also described by assigning values. Each vehicle can also be assigned to one of the available pollutant or noise emission classes. Various GUI parameters are also possible to assign. (Spaho et al., 2011)

Vehicles data can be created by various format of different application. For large application generally “origin/destination matrices” are used. Vehicle movement is specified in this between different areas at all moments. Generally, a single matrix data is not enough for result in traffic simulation as the direction changes over time. SUMO has features to convert OD matrices to its own single trip with the use of od2trips application. Vehicles departure and arrival movement can be specified by this application. The maps are given as input data to this application. The trips consist of a start and end road together with a departure time without explicit route data.

Routing Path between source and destination node is generated using various calculation based on various procedure after assigning traffic pattern. JTRROUTER us an application for finding route based on turn probability at junction point of network. This application is used specifically at city area with few vehicles traffic. DFROUTER is another application to search route by using data from loop detectors. This application is useful for areas like highway where road doesn’t have ring and it could be possible to fix start and end of road to fix detector. So it will fail on city network because it is not possible to fix detectors in city because of loop road.
Figure 4.1 Map Looks on Openstreet Site
Simulation is performed in a specified time with minimum simulation time of one second. Each vehicle is travelling on a fixed lane and position is described by its first position. Various car following models are used in network which describes speed of vehicles. SUMO uses an extension of the car-following model. Lane changing is only possible with use of various models design during implementation. Traffic simulation is achieved in SUMO using two ways. First with the use of command prompt and second with use of graphical user interface (GUI) based application using openGL. (Behrisch et al., 2011)

Different output will be generated on each simulation run. Vehicles position is written with use of various formats to define trips and condition of road traffic at a particular time. SUMO also supports various other models line noise emission, pollution emission and fuel consumption.

TraCI is one of the applications which extend functionality of SUMO by third party. SUMO Traffic Modeler used to generate a traffic density for a given map
which can be used later on to generate various traffic simulations to generate different
results. (Piorkowski et al., 2008) ACTIVITYGEN is another application which
generates different types of traffic for SUMO simulation. EWorld allows defining
different condition of environments like weather situation and its visualization during
simulation. (Cabrera et al., 2009)

SUMO is being used by various researchers which were discussed earlier in
literature review to solve various research problems which are mentioned as below.

1. Performance measurement for traffic lights to be performed base on weekly plan
for evaluation of different algorithms.

2. Nodes movement can be captured and studied during development of new
algorithms which can be used for eco friendly routing of information. This can
improve pollution emission of vehicles and how to choose autonomous route for
vehicles.

3. Traffic forecast for higher authorities is also possible with use of SUMO to
avoid future traffic problem.

4. Vehicles movement is useful for SUMO for performance of various application
based on GSM structure.

5. Both real vehicles travelling and actual simulation of vehicles can be easily
provided by SUMO vehicular simulator which can be used by the V2V and V2I.

Majorly two kinds of things keep ready before starting with SUMO traffic
simulation.

1. A Topology of Network and

2. Required Traffic Scenario
SUMO network topology is consist of different types of roads, pedestrian passing way, moving vehicles like cars, truck, buses, people etc.

All these components required in network topology are not useful unless it is assigned motion pattern with network topology. Both of these things can be supplied as configuration file in SUMO to execute simulation. Configuration can be created in terms of XML file.

<configuration>

<input>

<net-file value="gan1.net.xml"/>

<route-files value="gan1.rou.xml"/>

<additional-files value="gan1.add.xml"/>

</input>

</configuration>

Network topology is defined in network file (*.net.xml) while all required routing files are defined in routing files (*.rou.xml) which describe the vehicles traffic movement in terms of nodes (*.rou.xml). If additional component required which is not mentioned than that can be defined in supplementary documents (*.add.xml). These documents are not obligatory and additions function are defined in it for the simulation, similar to e.g. movement of nodes in road loops or different types of sensors fixed in the system, a background figure of aerial photos of that region which was used in simulation by giving nice and accurate visualization of map, Points Of Interest, like house names, shopping centre, or other milestone points which you would like to be included in it of the simulation.
Following command is used to start simulation of network in SUMO by configuration file gan1.sumocfg in command prompt.

sumo -c gan1.sumocfg

or

sumo-gui -c gan1.sumocfg

Network file defines traffic associated components of a areas where lots of vehicles can pass through. This network is to be defined as per traffic pattern. So it is a directed graph. All edges are unidirectional and have much information as mentioned below. (Behrisch et al., 2011)

1. Roads, streets, routes are always defined as a collection of edges connected with other edges. It has multiple lanes with different shape, position and speed limit.

2. A collection of vehicles along with different other components specify correct way of using junction.

3. A junction builds to control traffic with a set of traffic light.

4. All lanes are connected with each other at junction points. Various lane changing movements also specified in this file.

Various options which can be used in this file are also possible as mentioned in following.

<edge id="<ID>" from="<FROM_NODE_ID>" to="<TO_NODE_ID>" priority="<PRIORITY>"/>

<lane id="<ID>_0" index="0" speed="<SPEED>" length="<LENGTH>"
shape="0.00,495.05 248.50,495.05"/>
<lane id="<ID>_1" index="1" speed="<SPEED>" length="<LENGTH>" shape="0.00,498.35,2.00 248.50,498.35,3.00"/>

</edge>

<junction id="<ID>" type="<JUNCTION_TYPE>" x="<X-POSITION>" y="<Y-POSITION>"
incLanes="<INCOMING_LANES>" intLanes="<INTERNAL_LANES>"
shape="<SHAPE>"
request index="<INDEX>" response="<RELATIVE_MAJOR_LINKS>"
foes="<FOE_LINKS>" cont="<MAY_ENTER>"/>

......

</junction>

<tlLogic id="<ID>" type="<ALGORITHM_ID>" programID="<PROGRAM_ID>"
offset="<TIME_OFFSET>">

<phase duration="<DURATION#1>" state="<STATE#1>"/>

......

</tlLogic>
4.1.2 NS2

NS2 stands for Network Simulator Version 2. NS2 is an open-source software simulation with design of event-driven designed for research in network. It was developed in 1989. It is being used in industry, academic and government organization due to its popularity. Lots of research work was carried out constantly from last few years which results in development of various modules for routing, transport layer protocol, and application. Scripting language supported by NS2 is very easy for researcher to do research work in network to obtain various results generated by NS2. Definitely, NS2 has become the most widely used simulator amongst all existing simulator. All researchers need different function which is much more than existing NS2 simulators available modules. To obtain such kinds of functionality, one needs to understand NS2 architecture. Online tutorials always help us to understand working of NS2 simulator. With helps of such material it is possible to extend existing modules in NS2 for better results. (Spaho et al., 2011)

Ever since, several revolutions and revisions have marked the growing maturity of the tool, thanks to substantial contributions from the players in the field. Among these are the University of California and Cornell University who developed the REAL network simulator, the foundation on which NS is invented. Since 1995 the Defense Advanced Research Projects Agency (DARPA) supported the development of NS through the Virtual Inter Network Test bed project. Currently the National Science Foundation has joined the ride in development. (Reddy & Reddy, 2006)

NS2 architecture is shown in Fig. 4.3. It provides substantial support to simulate bunch of protocols like TCP, FTP, UDP, https and DSR. It simulates both wired and wireless network. It uses TCL as its scripting language. NS2 consists of two key languages: C++ and Object-oriented Tool Command Language (OTcl). While the C++ defines the internal mechanism (i.e., a backend) of the simulation objects, the OTcl sets up simulation by assembling and configuring the objects as well as scheduling discrete events. The C++ and the OTcl are linked together using TclCL. NS2 uses OTcl to create and configure a network, and uses C++ to run simulation. We use C++ when we are dealing with a packet, or when we need to modify existing
NS2 modules. All C++ codes need to be compiled and linked to create an executable file. In-built GEDIT is used to create or edit C++ file. NS2 simulator is available for various operating systems like UNIX (or Linux), Windows, and Mac systems. (Cabrera et al., 2009)

![Figure 4.3 NS2 Architecture](image)

Fig. 4.3 shows the directory structure of NS2 simulator under directory nsallinone-2.35. NSALLINONE is joint package of all required software within one directory. Level one contains directory nsallinone-2.35. On the Level two have directories like tclcl, tcl, tk and ns2.35. All NS2 modules required in simulation are available in the directory ns-2.35 on the Level two. TCL directory at Level 3 is one of the most used directories amongst all. It has lots of files which are used for simulation configuration like ns-lib.tcl, ns-node.tcl, ns-link.tcl etc. stored under subdirectory lib at Level 4. Tools directory contain various helper classes such as random variable generators. Directories queue, tcp, and trace contain modules for queue, TCP, and tracing, respectively. (Reddy & Reddy, 2006)
4.1.3 PYTHON

Python is a widely used general-purpose, high level programming language. It was mainly developed for emphasis on code readability, and its syntax allows programmers to express concepts in fewer lines of code. Python is a programming language that lets you work quickly and integrate systems more efficiently. Some of python scripts in SUMO are not possible to run without it. (Krajzewicz et al., 2012)

Python can be used for a text processing, system administration and internet related tasks. Its language is very easy to understand to become master in developing complex program which is not possible in any other programming language. Python is a true object-oriented language. It is also available on a different operating system. Python code can be used for solving various internet based problems because of its Java based interpreter. (Krajzewicz et al., 2012)

Python was developed initially during early 1990’s then at Centrum Wiskunde and Informatica in Amsterdam, and currently it is maintained and developed at Virginia. Later on python started to be used as one of computer programming language which is easy to learn and become advance programmer. So
Python is now becoming first choice for programmer without wasting much energy on programming learning. (Santana et al., 2015)

Python is available with some freedom while doing programming as well as strict to follow certain rules. C++ and Java are also object oriented programming language by design. Few more languages are also act as object oriented but they are not performing it from initial development as python is. (Santana et al., 2015)

4.1.4 PERL

Perl stands for “Practical Extraction and Report Language” or “Pathologically Eclectic Rubbish Lister.” Perl is an interpreted programming language. Perl is an open source language initially developed for word handling, system management, web development etc.

Perl supports different programming language structure. Various critical projects are also developed in it around all sectors. Perl is licensed under GNU General Public License (GPL). Perl was first developed by Larry Wall. (Santana et al., 2015)

Various programming languages good functionality is achieved with use of Perl. These different languages are like C, AWK, SED, SH, and BASIC. Different types of database engines are also integrated with Perl’s database engine. These databases are SQLPlus, SQLServer and others. Perl works with different types of markup languages like HTML, XML etc. Perl supports various types of text encoding like Unicode. More than 20000 modules of different developer are supported by Perl programming language with development of both procedural and object-oriented programming in Perl Archive Database. The Perl interpreter can also be implanted into other programming languages or framework.

Perl’s code can run without compiling it as it is an interpreted language. All compilers first convert program into machine code. While Perl programs directly run it and convert it into byte code then it will be transferred as machine code. So Perl is
not identical as shells, or TCL. Shells and TCL are rigorously interpreted exclusive of further illustration. (Santana et al., 2015)

It is not similar to C or C++, whose compiler straightforwardly converts its code into a machine code. So Perl comes between Python, AWK and Emacs files. Originally Perl was developed to configure and control some specific application. But later on as it grew it becomes general tool for system administration on different operating system. It then becomes famous amongst group of developers.

Perl was initially developed on UNIX platform. Currently Perl is available on more than 70 different platforms which include Windows, MacOS, QNX, BeOS, UNIX, LynxOS, VMS and Linux.

Various uses of Perl are listed below.

1. System administration
2. Numeric and text processing.
3. Database interconnectivity
4. CGI/Web programming
5. Useful in different programs like FTP, WWW, etc.

Perl provides three important qualities of a programmer as mentioned below.

1. Idleness – Problem solution is developed in terms of generic solution and reusable code development.
2. Eagerness – Various problem as per our need can be solved by developing it.
3. Hubris – Its program is easy to learn and write by other to maintain it for future development.

4.1.5 MOVE

MOVE (MObility model generator for VEhicular networks) is used to facilitate users to rapidly generate realistic mobility models for VANET simulations. Our tool MOVE is built on top of an open source micro-traffic simulator SUMO. The output of MOVE is a mobility trace file that contains information of realistic vehicle movements which can be immediately used by popular simulation tools such as ns-2 or qualnet. In addition, MOVE provides a set of Graphical User Interfaces that allows the user to quickly generate realistic simulation scenarios without the hassle of writing simulation scripts as well as learning about the internal details of the simulator. Implementation of MOVE is in Java and runs for SUMO traffic simulator. It has two major components like Map editor and Vehicle Movement Editor as shown in Fig. 4.5. The Map Editor is used to create the road topology. There are three different approaches are possible in developing road map. (Moustafa & Zhang, 2009)

1. Manual setup of file by user

2. Automatic generation

3. Imported from existing real world maps like TIGER and Open street map. TIGER stands for Topologically Integrated GEographic Encoding and Referencing. It’s a legal ma database of U.S.

The Vehicle Movement Editor allows us to generate various trips of vehicles and the route on which trip will be carried out by each vehicle. Vehicles movement can be generated by three different approaches as mentioned below. (Karnadi et al., 2007)

1. User can generate it manually

3. Depending on some fixed movements like public transportations like bus or train.

Figure 4.5 Mobility Model Generator
All maps and vehicles movement on different route generated by MOVE will be given as input into SUMO to generate trace file. This trace file can directly being used by wither ns2 or qualnet network simulator for real time results. (Karnadi et al., 2007)

### 4.1.6 GCC (GNU Compiler Collection)

GNU C Compiler was developed by Richard Stallman who was founder of GNU project. This project is started in 1984 to develop freeware software like UNIX operating system. This project success is achieved by cooperation of various users and developers from all around world. Initially free C compiler was not available in UNIX operating system. So GNU Project has started to develop C compiler from scratch. (Kumar et al., 2015)

Free Software Foundation is a non-profit organization who accepts donations from different organizations and users to develop different project. GCC was fully developed in 1987. This is crucial achievement towards developing free portable ANSI C compiler. From that time GCC has become most widely used tool in developing free software. Its development is guided by the GCC Steering Committee, a group composed of representatives from GCC user communities in industry, research and academia. (Kumar et al., 2015)

Various features of GCC are listed below.

GCC runs on various operating systems as well as produce output on many types of processor so it is portable compiler. It also runs on different microcontroller and 64 bit machines.

GCC supports native compiler. It also convert program into executables files which can be used in different operating system. This feature is known as cross compilation. It has additional functionality of generating code for embedded system because embedded system is not able to run a compiler. GCC compiler is coded in C
GCC supports generation of many programming languages architecture cross compilation code with its own front end. Like an ADA program can be compiled for a other microprocessor, or a C program for a supercomputer. (Kumar et al., 2015)

GCC design is modern which can adapt to new architectures and programming languages. If any new languages are added to GCC front end then architecture of those languages will be available at front end of GCC compiler with availability of additional run time facilities of libraries. Similarly, adding support for a new architecture will enable use of it for all languages. (Mittal & Choudhary, 2014)

Finally GCC is free software and licensed under GPL. So GCC is available for all users to use and modify. If any processor is not supported by GCC compiler than user is allowed to modify its code themselves which can support that processor. You can hire other programmer also to solve bugs. (Mittal & Choudhary, 2014)

All users are allowed to use as well as share updated copy of GCC compiler. Same way freedom is also given to use updated copy developed by other user also.

4.2 Traffic Model Creation Process

First open chrome or any other compatible browser then type http://www.openstreetmap.org following press enter button. Find out area which is required for simulation and then click on export button which is available on top of website, which result in downloaded file “map.osm”. Rename this file as per user requirement.

Maps are derived from internet map database source like open street or tiger. Both maps for rural (Fig. 4.6) and urban (Fig. 4.7) are used for simulation.
To achieve result we need to open terminal and go to SUMO directory and type various commands as mentioned below.

```
netconvert --osm-files map.osm -o gan1.net.xml
```

Figure 4.6 Rural Maps
Run this command to get the .net file required for simulation. Netconvert imports digital road networks from different sources and generates road networks that can be used by other tools from this package. It stores the SUMO-network generated from this data into "gan1.net.xml".

From the following link copy the additional polygons structures and save it as gan1.typ.xml.

http://sumo.dlr.de/wiki/Networks/Import/OpenStreetMap

Then run following command.

Figure 4.7 Urban Maps
polyconvert --osm-files map.osm --net-file gan1.net.xml --type-file gan1.typ.xml -o gan1.poly.xml

Polyconvert imports geometrical shapes (polygons or points of interest) from different sources, converts them to a representation that may be visualized using SUMO-GUI.

python /home/Hitesh/sumo/tools/randomTrips.py -n gan1.net.xml -r gan1.rou.xml -e 100 -l

python /home/Hitesh/sumo/tools/randomTrips.py -n gan1.net.xml -r gan1.rou.xml -e 100 -l

“randomTrips.py” generates a set of random trips for a given network (option -n). It is possible by selection source and destination edge with uniformly at random or biased by length, by number of lanes or both. The resulting trips are stored in an xml file like *.trips.xml which is suitable for the DUAROUTER by calling automatically using -r option is given. Entire trips depend on time span mentioned as starting and ending time in seconds. All trips can be repeated as specified in seconds. Each trip has unique id with different prefix and its travelling number.

After the above steps, the routes have been generated, in the xml file you need to configure the file for the SUMO-GUI. Now, configuration file is required to be created by typing following lines in it.

<configuration>

<input>

<net-file value="gan1.net.xml"/>

<route-files value="gan1.rou.xml"/>

<additional-files value="gan1.poly.xml"/

58
Above configuration file can be loaded using sumo-gui gan1.sumo.cfg and output of it is as per following screenshot in Fig.4.8.

Figure 4.8 Sumo GUI Output
4.3 NS2 Simulation steps

Now sumo trace needs to be converted to ns2 file. Run following code for this.

```
sumo -c gan1.sumo.cfg --fcd-output gan1.sumo.xml
```

This fcd output command creates a trace file in a SUMO-format. This file will be later converted into a trace file for one of the applications supported by Tools/TraceExporter.

```
python /home/Hitesh/sumo/tools/traceExporter.py --fcd-input gan1.sumo.xml --ns2config-output gan1.tcl --ns2activity-output gan1act.tcl --ns2mobility-output gan1mob.tcl
```

Output of above commands will result in three TCL files like gan1.tcl, gan1act.tcl and gan1mob.tcl. Out of all these files second file is not required in our simulation. Third file is compulsory required for ns2 simulation. First file generated is gan1.tcl. it will be changed according to various network parameters like routing protocol, Mac layer, number of nodes, different layer data etc. Sample TCL script file gan1.tcl for ns2 simulation is as shown below. Mobility.tcl file is taken as traffic routing file.

```
set val(wch)           Channel/WirelessChannel
set val(trg)           Propagation/TwoRayGround
set val(wlp)          Phy/WirelessPhy
set val(m)            Mac/802_11
set val(intf)            Queue/DropTail/PriQueue
set val(link)             LL
```
set val(omn) Antenna/OmniAntenna

set val(intfl) 50

set val(non) 41

set val(pro) MCFS

set opt(x) 4707

set opt(y) 3002

set ns_ [new Simulator]

set tracefd [open gan1.tr w]

$ns_ trace-all $tracefd

set namf [open gan1.nam w]

$ns_ namtrace-all-wireless $namf $opt(x) $opt(y)

set tog [new Topography]

$topo load_flatgrid $opt(x) $opt(y)

create-god $val(nn)

$ns_ node-config -adhocRouting $val(pro) \
    -llType $val(link) \
    -macType $val(m) \
    -ifqType $val(intf) \n
-ifqLen $val(intfl) \\
-antType $val(omn) \\
-propType $val(trg) \\
-phyType $val(wlp) \\
-channelType $val(wch) \\
-topoInstance $tog \\
-agentTrace ON \\
-routerTrace ON \\
-macTrace OFF \\
-movementTrace ON \\

for {set j 0} {$j < $val(non)} {incr j} {

    set node_($j) [$ns_ node] 
    $node_($j) random-motion 0 
    $ns_ initial_node_pos $node_($j) 30 
}

source gan1mob.tcl

set tcp [new Agent/TCP]

$tcp set class_ 2
set sink [new Agent/TCPSink]

$ns_ attach-agent $node_(5) $tcp

$ns_ attach-agent $node_(34) $sink

$ns_ connect $tcp $sink

set ftp [new Application/FTP]

$ftp attach-agent $tcp

$ns_ at 10.0 "$ftp start"

for {set j 0} {$j < $val(nn) } {incr j} {
    $ns_ at 200.0 "$node_($j) reset";
}

$ns_ at 200.0 "End"

$ns_ at 200.01 "puts "Over.."

proc stop {} {
    global ns_ tracefd

    $ns_ flush-trace

    close $tracefd
}

puts "Start of Program...."

$ns_ run
4.4 MCFS Protocol Files

Sumo tool is used to create vehicle movement according to map’s road. Node movement is not possible to achieve according to road in NS2 because all nodes moves in random position. NS2 is used to run various protocols on road generated with the use of SUMO. Few protocols are readily available in NS2. But here new protocol design is proposed which is better than already existing protocol. This proposed protocol along with existing protocol is tested on different parameter.

Proposed protocol is implemented with concept of C++. MCFS protocol implementation is divided into various files according to its functionalities. All files are listed below with its functionalities.

MCFS.cc

This is main implemented file of MCFS protocol. This file has almost all function implementation for MCFS protocol. Various functions provided by this file includes initialization, message handling, beacon timer, sending and receiving of packet, source and destination node position, get neighbor data, finding route of packet and receive signal.

MCFS.h

This file is a main header file for MCFS protocol. It is working as interface file for various function implemented in MCFS protocol. All methods prototype definition is declared in MCFS protocol.

MCFS.msg

This file is used to send beacon signal in regular interval to search path from source node to destination node.

MCFS.ned
This file is basic interface module of MCFS protocol for wireless networks.

MCFSDefs.h

This file is routing interface file for MCFS protocol.

PositionTable.cc

This file is used to maintain point table for MCFS protocol. It has implementation of many functions like get position, set position and remove position of nodes during maintaining Position table of all nodes.

PositionTable.h

This file is header file for PointTable.cc file. It is also known as interface file for creating point table.