**The OMNeT**



**Written by Hwanseok Jeong in SKKU  
, Sangsoo Jeong in DGIST**

**Index**

1. **Time Table - 3**
2. **Installation Guide - 4**
   1. **OMNeTpp - 4**
   2. **Sumo - 7**
   3. **Veins - 8**
3. **Practice of OMNeT Example - 11**
   1. **Tictoc - 11**
4. **Current Work - 17**

**[ Time schedule for OMNeT++ Simulator ]**

■ When: 09:00~12:00 AM on Saturday on Feb. 15, in 2014 in Korean Time(7PM on Feb. 14 in EST)

■ Where: CPSLab in SKKU and on SKYPE

■ Participants:

**1. Requirement to Install OMNET++, SUMO, Veins**

● Installation document of OMNeT++, SUMO, Veins is released until this weekend. And participants can install and test OMNeT++, SUMO, Veins installation guide before this seminar on Feb. 15.

● Pre-installation and test of simulator are important to be along with this seminar.

**2. Time Table for Seminar**

● Time Table

|  |  |  |
| --- | --- | --- |
| Time line | Contents | Speaker |
| 09:00 ~ 09:30  (30 min) | Brief explanation for Installation and architecture of simulator   * Installation guide * Overall architecture of OMNeT++, SUMO * Relation with OMNeT++, SUMO, Veins | Sangsoo Jeong |
| 09:30 ~ 09:40 (10 min) | Break Time |  |
| 09:40 ~ 10:40  (60 min) | OMNeT Samples and construction the OMNeT project with this sample   * Sample OMNeT++ Project: Tictoc, Autoconf * Step by step Practice with this sample project * QnA | Sangsoo Jeong |
| 10:40 ~ 10:50 (10 min) | Break Time |  |
| 10:50 ~ 11:50  (60 min) | Structure and implementation of WPCF and TMAC in OMNeT   * WPCF architecture and source code in OMNeT++ * TMAC architecture and source code in OMNeT++ * QnA | Sangsoo Jeong |
| 11:50 ~ 12:00 (10 min) | Wrap-up |  |

# There may be trivial changes in time schedule according to the particular situation.

**1. Installation Guide**

**Both Windows & Linux support OMNeT & Sumo but this document is for Ubuntu**

1. **OMNeTpp**  
   OMNeT is C++ based network simulator and it can be run on Eclipse. It is very helpful for researcher on network field, because it supports animation of packet flow on network and evaluation of network performance
   1. **Installation of OMNeT**
      1. **Download OMNeTpp**  
         [http://OMNeTpp.org/component/docman/cat\_view/17-downloads/1-OMNeT-releases](http://omnetpp.org/component/docman/cat_view/17-downloads/1-omnet-releases)
      2. **Decompress OMNeTpp**  
         $ OMNeT++$ tar zxvf OMNeTpp-4.3.1-src.tgz  
         $ sudo mv -v OMNeTpp-4.3.1 /usr/local/src
      3. **Configure and Install**  
         $ cd /usr/local/src/OMNeTpp-4.3.1  
         $ ./configure  
         $ make
      4. **Register on environment variable**  
         $ PATH=/home/USERNAME/Simulation/OMNeT++/OMNeTpp-4.3.1/bin:$PATH  
         $ export PATH  
         $ export TCL\_LIBRARY=/usr/share/tcltk/tcl8.5
      5. **Download Java**<http://java.com/ko/download/linux_manual.jsp?locale=ko>
      6. **Decompress Java and move**$ OMNeT++$ tar zxvf jre1.7.0\_45.tgz  
         $ mv jre1.7.0\_45 /usr/local/src/OMNeTpp-4.3.1/ide/jre
      7. **Start**  
         $ OMNeTpp
   2. **A structure of OMNeT**  
      The OMNeT is operated by interaction of three files. Below figure1 illustrate this operation.

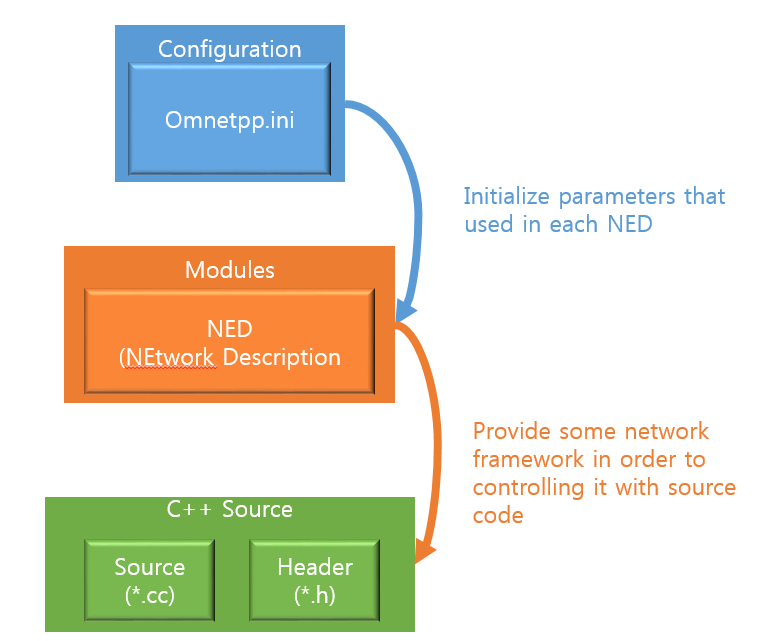


Figure 1 Diagram of OMNeT components

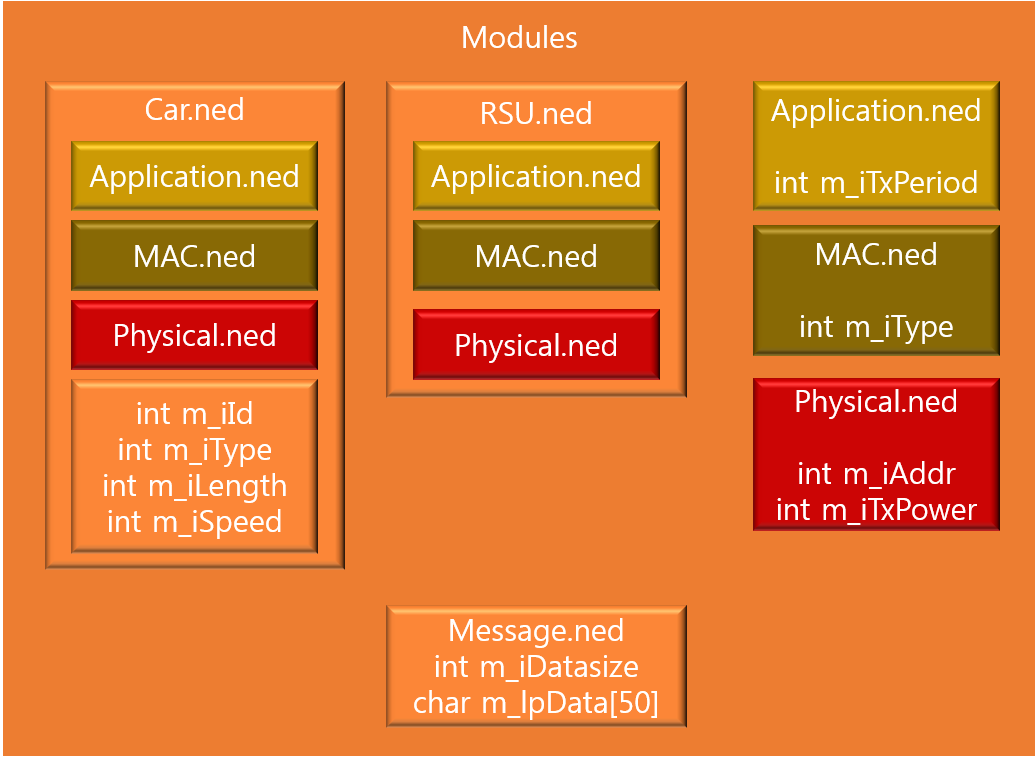
* + 1. **Description**
       1. **OMNeTpp.ini**This ini file selects a NED file about scenario that you want to simulate and initialize parameters used in NED file.
       2. **NED**The scenario that you want to simulate should be designed in this NED file. Also, NED object should be define in NED file. This NED object is very similar to class of C++ and Java. Once NED object is created, it can be inherited by other NED object. i.e. if B.ned inherit A.ned, B can use all of attributes and functions on A. Figure2 shows this relation.  
          

Figure 2

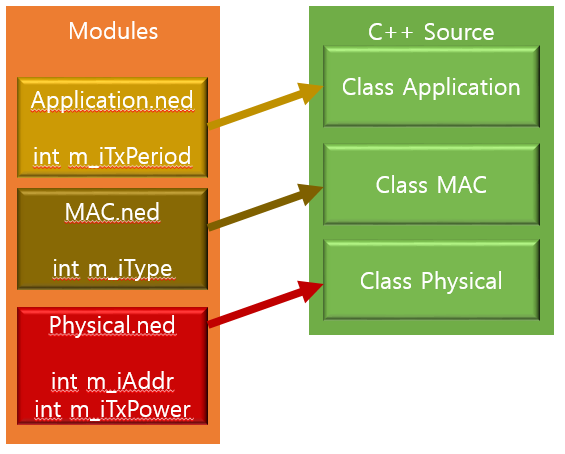
* + - 1. **C++ Source**NED is network design by deploying NED objects but it can’t control each object. So, to control objects, OMNeT support c++ source code. To do that, NED object should be connected with class. After connection is established, source code in class will work and control all objects. The diagram of it is described in figure 3.

Figure 3

1. **Sumo**
   1. **Installation of Sumo**
      1. **Update apt-get command**   
         $ sudo apt-get update  
         $ sudo apt-get upgrade  
         $ sudo apt-get install aptitude  
         $ sudo aptitude install bison flex build-essential zlib1g-dev tk8.4-dev blt-dev libxml2-dev sun-java6-jre libpcap0.8-dev autoconf automake libtool libxerces-c2-dev proj libgdal1-dev libfox-1.6-dev.
      2. **Symbolic link**   
         $ sudo ln -s /usr/lib/libgdal1.7.0.so /usr/lib/libgdal.so
      3. **Download Version: 0.15 (Statable Version) at SUMO official site**Official site : <http://sumo-sim.org/wiki/Downloads>  
           
         0.15 Version :   
         <ftp://ftp.mirrorservice.org/sites/downloads.sourceforge.net/s/su/sumo/sumo/version%200.15.0/sumo-src-0.15.0.tar.gz>  
           
         0.19 Version : <http://prdownloads.sourceforge.net/sumo/sumo-src-0.19.0.tar.gz?download>
      4. **Decompress SUMO installation package and move it to System Source Directory**$ tar -xzvf sumo-src-0.15.0.tar.gz  
         $ sudo mv -v sumo-0.15.0 /usr/local/src  
         $ cd /usr/local/src/sumo-0.15.0
      5. **Configure**$ ./configure --with-fox-includes=/usr/include/fox-1.6 \--with-gdal-includes=/usr/include/gdal --with-proj-libraries=/usr \--with-gdal-libraries=/usr --with-proj-gdal --enable-debug

// if an error about fox-1.6 occur then you can use **sudo apt-get install fox-1.6** then step ‘E’ will be completed success

* + 1. **Install**$ make  
       $ sudo make install
    2. **Call SUMO and Check**$ sumo-gui  
       Or  
       $ sumo  
         
       If you type sumo-gui, you run gui version of sumo. On the other hand, if typing sumo, command line mode is run.

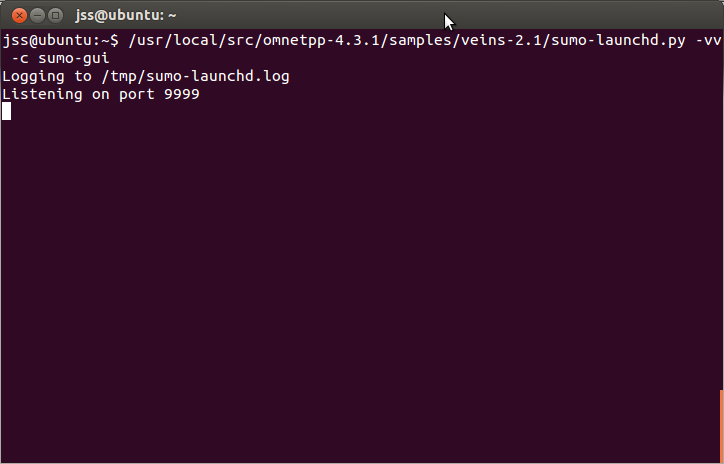
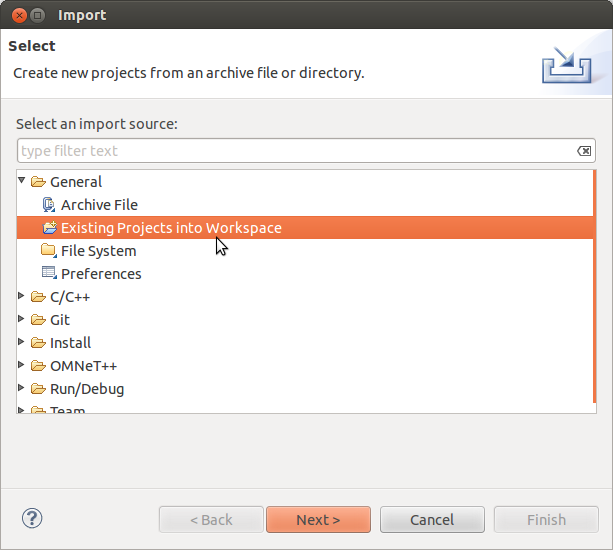
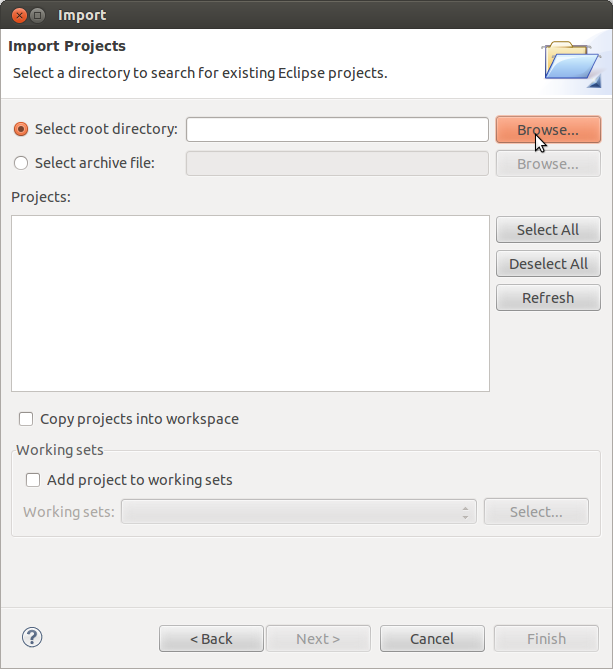
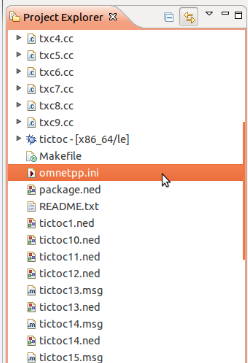
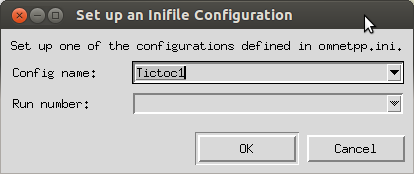
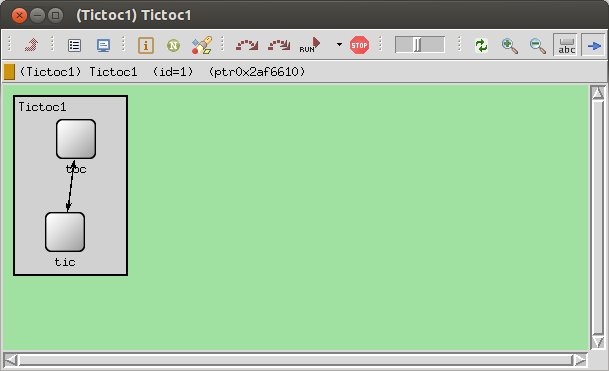
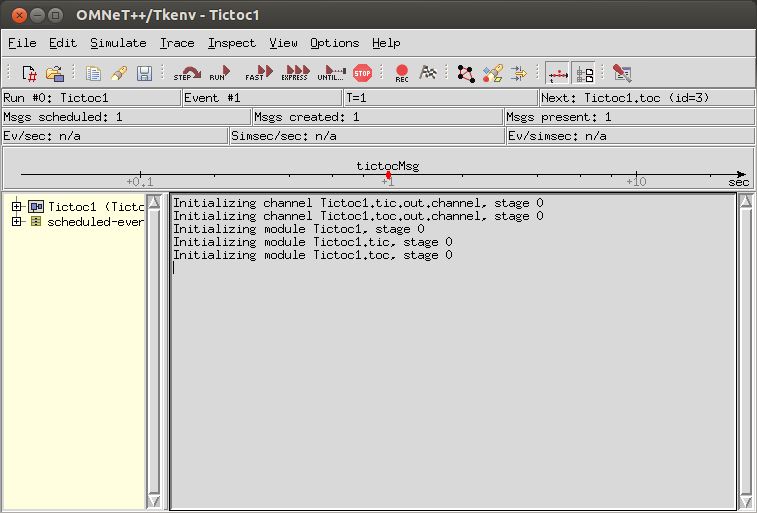
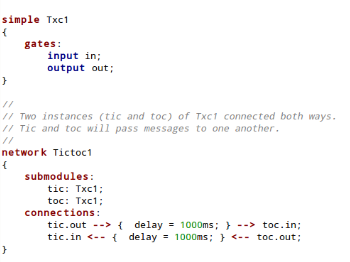
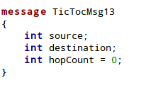
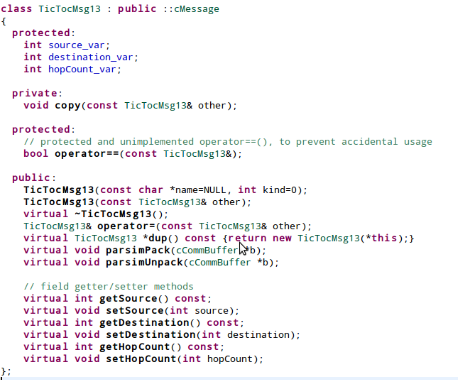
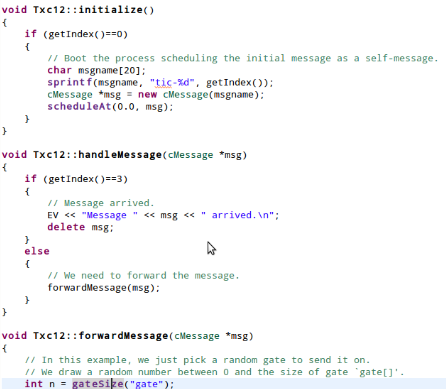
1. **Veins**Veins is advanced framework which is generated using basic OMNeT functions so usage is same as original OMNeT. As it makes connection between OMNeT and sumo, it combines benefit of both simulator.
   1. **Installation of Veins**
      1. **Download Veins**  
         <http://veins.car2x.org/download/>
      2. **Follow the step**<http://veins.car2x.org/tutorial/>  
         But if both Sumo and OMNeT is installed on your computer, then skip all the step except final step. The next is core contents of final step.  
           
         $ <VEINS\_INST\_DIR>/sumo-launchd.py –vv –c sumo-gui  
         Then terminal will be blocked and it will wait the connect request of OMNeT.  
         

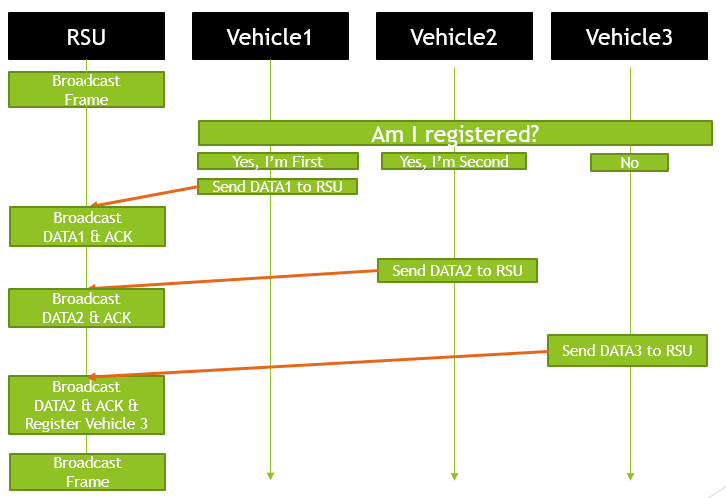
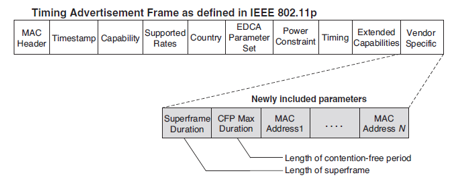
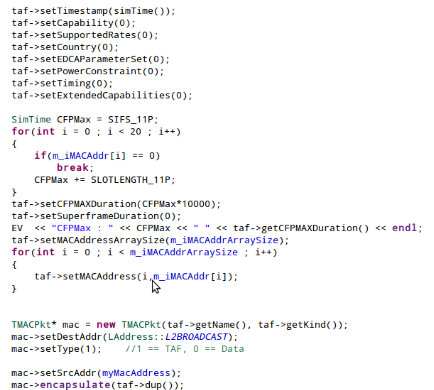
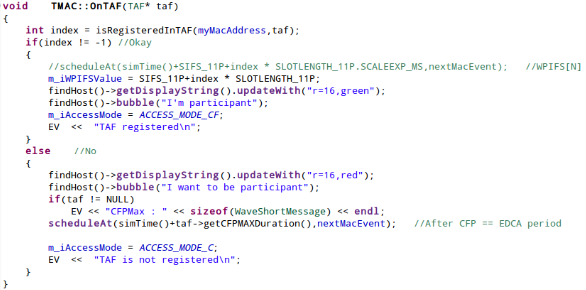
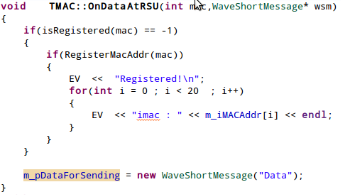
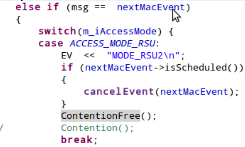
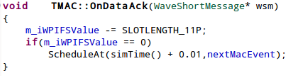
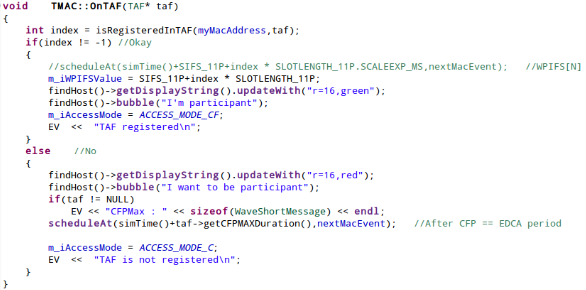
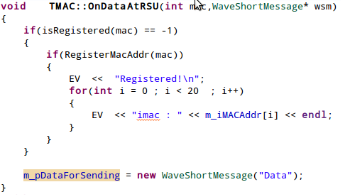
Figure 4 Success to run

* + 1. **Run the veins project**After execution of OMNeT, find existing project ~~~ on File – import menu. And set the veins installation directory. If success, you can see mixim directory on you project window on Omne.> Then click mouse right button placing your mouse on mixim/samples/veins/OMNeTpp.ini and click “Run As OMNeTpp”. Then this veins project will be compiled and run.
  1. **A structure of Veins**
     1. **OMNeT aspects**Because the Veins is just framework on OMNeTpp, it follows rules of OMNeTpp. However instead of calculating position of vehicles on every tick, it receives vehicles information from Sumo through TCP/IP.
        1. **TraciScenarioManager**It makes a connection between Sumo & OMNeT. When you run **sumo-launchd.py** on your terminal, it will wait Veins’ connection, calling accept(). After that, if you run Veins, then TraciScenarioManager class call connect() and Sumo and Veins is connected.
        2. **omnetpp.ini**This file is configuration file which initializing all values used in modules on OMNeT. However in Veins, you should mention which sumo file you want to cooperate.
     2. **Sumo aspects**Because Veins uses Sumo, files for sumo should be placed in project directory (where OMNeTpp.ini exists).
        1. **<filename>.con.xml**Defining connection between two lanes.
        2. **<filename>.edg.xml**  
           Defining edge between two nodes.
        3. **<filename>.net.xml**  
           Defining formation of edge and its rules.
        4. **<filename>.nod.xml**Defining position of node.
        5. **<filename>.rou.xml**  
           Defining a car route and its generation.
        6. **<filename>.typ.xml**  
           Defining configuration of edge.
        7. **<filename>.sumo.cfg**  
           **<filename>.netccfg**  
           **<filename>.launchd.xml**  
           These three files are used for configuration and you should write which sumo file will you use.

**2. Practice of OMNeT Examples**

* 1. **Tictoc**
     1. If you are first on OMNeT, you should import some basic example  
        Menu -> File -> Import -> General -> Existing Project into Workspace -> Next
     2. Select root directory -> Browse -> <Find directory that smaples exist> -> Ok ->Finish 
     3. Find ‘tictoc’ in Project explorer and find omnetpp.ini.  
        Place your cursor on it -> Right click -> Run as -> Omnet++ simulation  
        
     4. In ‘config name’ combo box, you can choose various type of tictoc example.  
        Choose Tictoc1 and click OK.  
        ****
     5. We can see two windows. In the first window, you can see a movement of packet.  
          
        Second window shows status of simulation, such as log, time, parameter in message etc…  
        
     6. There is very similar toll bar on both window. If you click curved arrow button , your simulation run and process just one event but if you click run button , then your simulation run until program ends or you stop it.
     7. Run the project and see how it works.
     8. Look at the files in Tictoc project.
        1. **omnetpp.ini**It initialize all parameter used in omnet simulation.
        2. **\*.ned**It provides a simulation frame. i.e. It define & deploy modules.  
             
           ‘simple Txc1’ is definition of simple module. And ‘network Tictoc1’ is modules deployment of simulation environment.To communicate each other, each module need pair gate (input & output).
        3. **\*.msg**This file defines a message used for communication  
           
        4. **\*\_m.cc , \*\_m.h**These two files is automatically generated by \*.msg file when project is compiled.  
           You can use message class defined in this file. But it is not allowed to modify these files directly.  
           
        5. **\*.cc , \*.h**These files can control modules and make a connection to module defined in ned file  
           This connection between source code and modules in ned is necessary, and it can be done by type   
             
           in your source code.  
             
           

**Current Work**

1. **Structure of mixim**
   * + 1. **Connection Manager**
          1. This class stores nic (LAN card) information to allow host to communicate with each other
          2. When a host want to transmit data to the other, it measure a distance of these two hosts and decide whether send or not.
          3. It manages all of In & Out gate of all submodules in this network (including hosts).
       2. **Mobility**
          1. All hosts should have mobility modules. Using mobility module is the only way to register its nic to Connection Manager. (although you don’t want to move hosts, mobility is still needed but you set empty mobility class)
          2. If you want vehicle follow sumo’s mobility, use TraCIMobility, then it automatically bring car data from sumo.
       3. **Application layer**This module is the highest layer of mixim, so y ou can make vehicular packet (which is called WaveShortMessage), and just send down to MAC layer.
       4. **MAC layer**Data generated by Application layer will send down to physical layer through this layer.
       5. **Physical layer**Managing channel status, This is lowest layer of sending & receiving data sequence. In other words, this provide interface among hosts.
   1. **TMAC**
      1. **WPCF sequence**
         1. RSU broadcasts \*Timing Advertisement Frame (TAF) which contains superframe duration, Contention Free Period MAX Duration and vehicles’ MAC address. Here, the sequence of MAC address is decided in order of that who will first de part from Tx range of RSU.  
              
               
            -----------------------------  
            
         2. Once a vehicle receives TAF, a vehicle search its MAC address in TAF. If so, it calculates WPIFS values and waits to send data until WPIFS values becomes zero.  
              
            But if not, it waits that a transmission of all the other registered vehicles quits.   
              
            
         3. When RSU receives data from Vehicle, RSU broadcasts ACK to all vehicle in Tx range, then WPIFS value of each vehicle is reduced.  
              
              
            --------  
              
            ---------  
            
         4. After all the other end its transmission (i.e. after CFP MAX Duration), unregistered vehicle can send data in EDCA mechanism (contention-based protocol). If success to send data to RSU, RSU registers MAC address of this vehicle.  
              
              
            --------------  
            
         5. Another TAF is sent and all step is repeated.