OMNeT++ Introduction

Pedro M. Santos (pss@isep.ipp.pt)
CISTER/ISEP

2021/06/02

CISTER – Research Centre in
Real-Time & Embedded Computing Systems

Outline

1. Introduction to Discrete Event Simulators & OMNeT
2. Component-based Architecture
3. Inter-module communication
4. A bit more on modules
5. Object-Oriented Programming
6. Practical Aspects

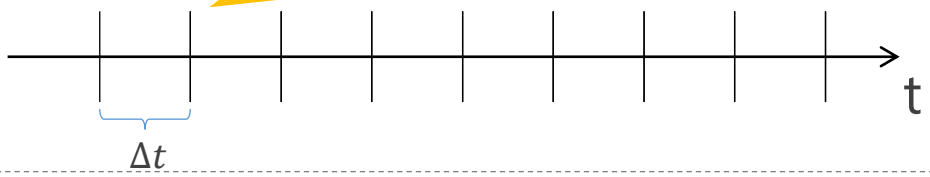
(Some bits inspired by Christian Timmerer's Slides "An introduction to OMNeT++ 4.2")

Discrete Event Simulator

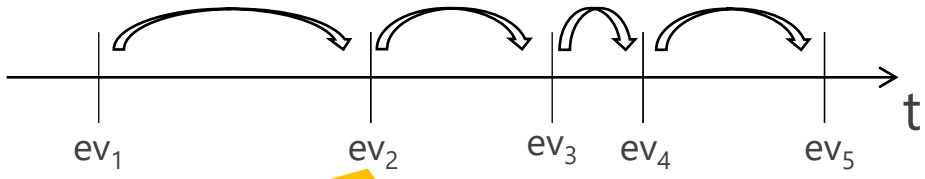
- Types of simulators
  - **Timestep:**
    - State of system and/or agents is evaluated every delta
    - Pros: easy to program
    - Cons: inferior temporal accuracy
  - **Discrete event:**
    - Events are scheduled for the future
    - Pros: more temporal accuracy
    - Cons: harder to program

Timestep:

System/agents evaluated every delta, whether there is or not a relevant update

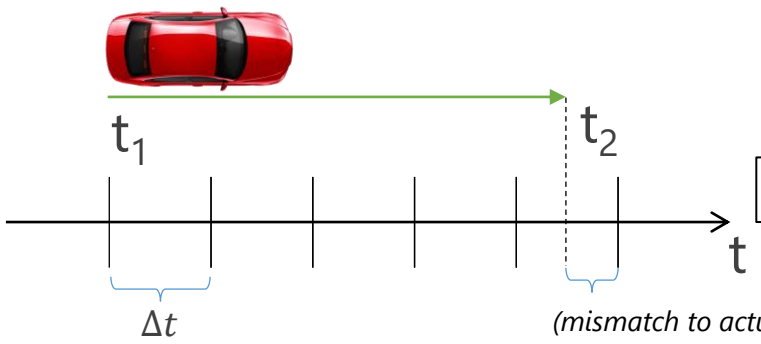


Discrete Event:



Only event instants are evaluated

Timestep:

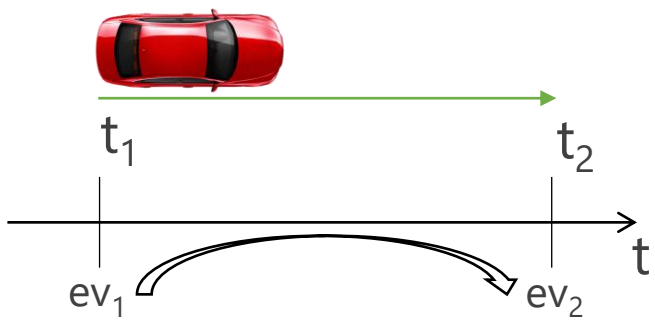


Evaluated at every Δt:

$$x_2 = x_1 + vt$$

(mismatch to actual event instant)

Discrete event:



Computing next event instant:

$$x_2 = x_1 + vt$$
$$t = (x_2 - x_1)/v$$

OMNeT++

- OMNeT is a discrete event simulator
- Gained popularity with networking simulation
- OMNeT follows a component (or module)-oriented paradigm
- Two main features:
  1. **Hierarchically nested modules**
  2. **Modules communicate using messages through channels**

1. Component-based Architecture

The NED Syntax

Component-based Architecture – The NED Syntax

- Network Description Language (NED): OMNeT-specific syntax to define modules and scenarios
- NED lets the user declare:
  - **Simple modules:** mapped into a .cc, .h and .ned files
  - **Compound modules:** mapped only into **.ned** file
  - **Networks:** mapped only into .ned file
- Compound and network modules are examples of **nested** modules
- Modules have:
  - **Parameters**
  - **Gates:** the mechanism through which modules send messages to each other
    - Very important for the 2nd feature ("*Modules communicate using messages through channels*")

Single Module

```
simple DemoBaseApplLayer like IBaseApplLayer
{
    parameters:
        @class(veins::DemoBaseApplLayer);
        int headerLength = default(88bit) @unit(bit); //header length of the application

        bool sendBeacons = default(true); //tell the applayer to periodically send
beacons
        int beaconLengthBits = default(256bit) @unit(bit); //the length of a beacon
packet
        int beaconUserPriority = default(7); //the user priority (UP) of the beacon
messages
        double beaconInterval = default(1s) @unit(s); //the intervall between 2 beacon
messages

        int dataLengthBits = default(1024bit) @unit(bit); //the length of a data packet
        bool dataOnSch = default(false); //tells the applayer whether to use a service
channel for datapackets or the control channel
        int dataUserPriority = default(7); //the default user priority (UP) for data
packets

        bool avoidBeaconSynchronization = default(true); //don't start beaconing directly
after node was created but delay to avoid artifical synchronization

        bool sendWSA = default(false);
        int wsaLengthBits = default(250bit) @unit(bit);
        double wsaInterval =  default(1s) @unit(s);

    gates:
        input lowerLayerIn; // from mac layer
        output lowerLayerOut; // to mac layer
        input lowerControlIn;
        output lowerControlOut;
}
```

projects - veins/src/veins/modules/application/ieee80211p/DemoBaseApplLayer.ned - OMNeT++ IDE

File Edit Source View Navigate Search Project Run Window Help

Car.ned | RSU.ned | Scenario.ned | omnetpp.ini | DemoBaseApplLay

package org.car2x.veins.modules.application.ieee80211p

DemoBaseApplLayer

projects - veins/src/veins/modules/application/ieee80211p/DemoBaseApplLayer.h - OMNeT++ IDE

File Edit Source Refactor Navigate Search Project Run Window Help

Car.ned | RSU.ned | omnetpp.ini | DemoBaseApplLay | DemoBaseApplLay

```
class VEINS_API DemoBaseApplLayer : public BaseApplLayer {

public:
    ~DemoBaseApplLayer() override;
    void initialize(int stage) override;
    void finish() override;

    void receiveSignal(cComponent* source, simsignal_t signalID, cObject* obj, cObject* details

    enum DemoApplMessageKinds {
        SEND_BEACON_EVT,
        SEND_WSA_EVT
    };

protected:
```

Writable | Smart Insert | 58 : 34 | Simulating veins (1) - Run : (0%)

projects - veins/src/veins/modules/application/ieee80211p/DemoBaseApplLayer.cc - OMNeT++ IDE

File Edit Source Refactor Navigate Search Project Run Window Help

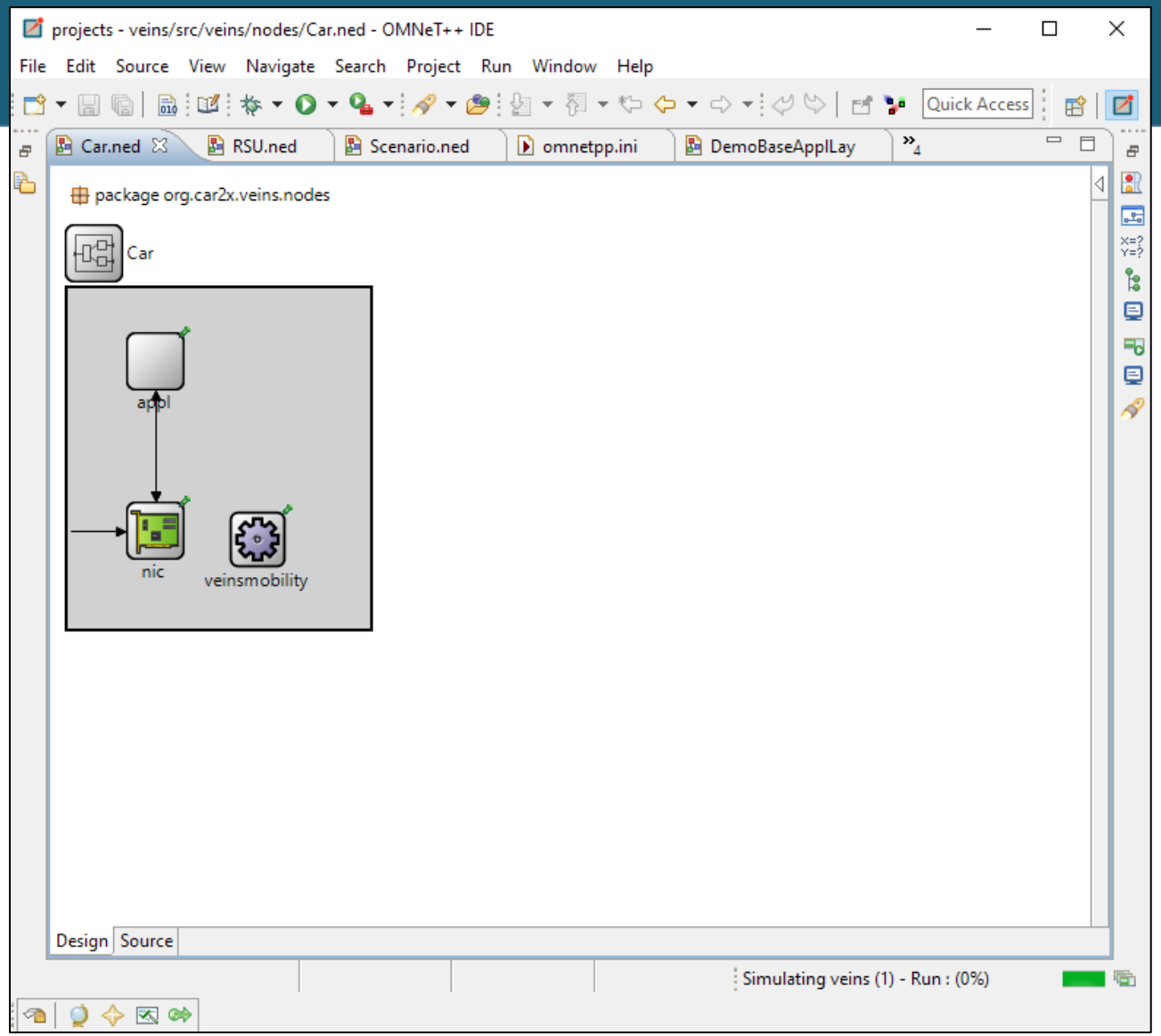
Car.ned | DemoBaseApplLay | DemoBaseApplLay | DemoBaseApplLay

```
void DemoBaseApplLayer::initialize(int stage)
{
    BaseApplLayer::initialize(stage);

    if (stage == 0) {

        // initialize pointers to other modules
        if (FindModule<TraCIMobility*>::findSubModule(getParentModule())) {
            mobility = TraCIMobilityAccess().get(getParentModule());
            traci = mobility->getCommandInterface();
            traciVehicle = mobility->getVehicleCommandInterface();
        }
        else {
            traci = nullptr;
            mobility = nullptr;
```

Writable | Smart Insert | 27 : 46 | Simulating veins (1) - Run : (0%)

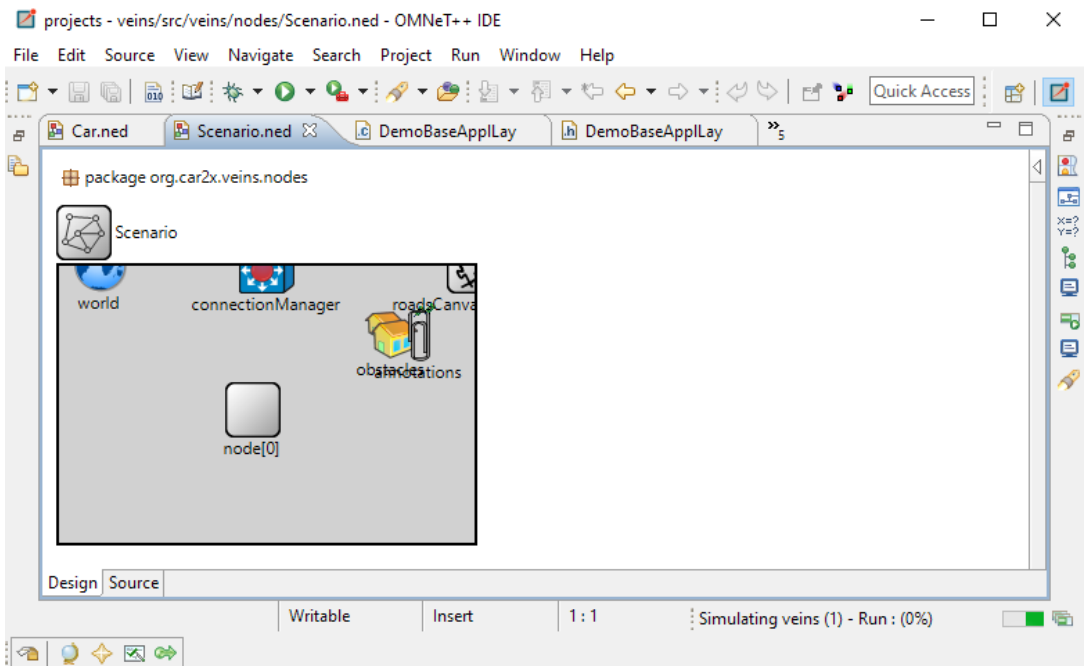
A Compound Module

```
module Car
{
    parameters:
        string applType; //type of the application layer
        string nicType = default("Nic80211p"); // type of network interface card
        string veinsmobilityType =
default("org.car2x.veins.modules.mobility.traci.TraCIMobility"); //type of the
mobility module
    gates:
        input veinsradioIn; // gate for sendDirect
    submodules:
        appl: <applType> like org.car2x.veins.base.modules.IBaseApplLayer {
            parameters:
                @display("p=60,50");
        }

        nic: <nicType> like org.car2x.veins.modules.nic.INic80211p {
            parameters:
                @display("p=60,166");
        }

        veinsmobility: <veinsmobilityType> like org.car2x.veins.base.modules.IMobility
{
            parameters:
                @display("p=130,172;i=block/cogwheel");
        }

    connections:
        nic.upperLayerOut --> appl.lowerLayerIn;
        nic.upperLayerIn <-- appl.lowerLayerOut;
        nic.upperControlOut --> appl.lowerControlIn;
        nic.upperControlIn <-- appl.lowerControlOut;

        veinsradioIn --> nic.radioIn;
}
```



A Network Module

```
network Scenario
{
    parameters:
        double playgroundSizeX @unit(m); // x size of the area the nodes are in (in meters)
        double playgroundSizeY @unit(m); // y size of the area the nodes are in (in meters)
        double playgroundSizeZ @unit(m); // z size of the area the nodes are in (in meters)
        @display("bgb=$playgroundSizeX,$playgroundSizeY");
    submodules:
        obstacles: ObstacleControl {
            @display("p=240,50");
        }
        annotations: AnnotationManager {
            @display("p=260,50");
        }
        connectionManager: ConnectionManager {
            parameters:
                @display("p=150,0;i=abstract/multicast");
        }
        world: BaseWorldUtility {
            parameters:
                playgroundSizeX = playgroundSizeX;
                playgroundSizeY = playgroundSizeY;
                playgroundSizeZ = playgroundSizeZ;
                @display("p=30,0;i=misc/globe");
        }
        manager: TraCIScenarioManagerLaunchd {
            parameters:
                @display("p=512,128");
        }
        roadsCanvasVisualizer: RoadsCanvasVisualizer {
            @display("p=300,0");
        }
        node[0]: Car {
        }

    connections allowunconnected:
}
```



2. Inter-module Communication

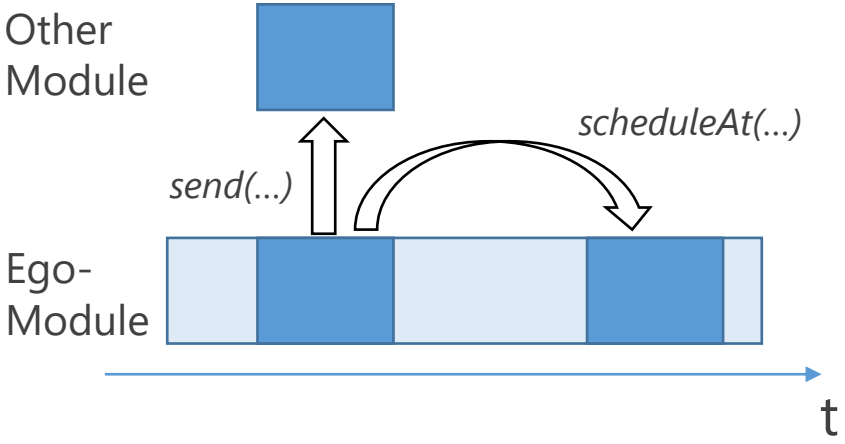
Gates and Connections

```
simple DemoBaseApplLayer like IBaseApplLayer
{
    parameters:
        @class(veins::DemoBaseApplLayer);
        int headerLength = default(88bit) @unit(bit); //header length of the application

        bool sendBeacons = default(true); //tell the applayer to periodically send
beacons
        int beaconLengthBits = default(256bit) @unit(bit); //the length of a beacon
packet
        int beaconUserPriority = default(7); //the user priority (UP) of the beacon
messages
        double beaconInterval = default(1s) @unit(s); //the intervall between 2 beacon
messages

        int dataLengthBits = default(1024bit) @unit(bit); //the length of a data packet
        bool dataOnSch = default(false); //tells the applayer whether to use a service
channel for datapackets or the control channel
        int dataUserPriority = default(7); //the default user priority (UP) for data
packets

        bool avoidBeaconSynchronization = default(true); //don't start beaconing directly
after node was created but delay to avoid artifical synchronization

        bool sendWSA = default(false);
        int wsaLengthBits = default(250bit) @unit(bit);
        double wsaInterval =  default(1s) @unit(s);

    gates:
        input lowerLayerIn; // from mac layer
        output lowerLayerOut; // to mac layer
        input lowerControlIn;
        output lowerControlOut;
}
```

```
module Car
{
    parameters:
        string applType; //type of the application layer
        string nicType = default("Nic80211p"); // type of network
interface card
        string veinsmobilityType =
default("org.car2x.veins.modules.mobility.traci.TraCIMobility"); //type
of the mobility module
    gates:
        input veinsradioIn; // gate for sendDirect
    submodules:
        appl: <applType> like org.car2x.veins.base.modules.IBaseApplLayer
{
            parameters:
                @display("p=60,50");
        }

        nic: <nicType> like org.car2x.veins.modules.nic.INic80211p {
            parameters:
                @display("p=60,166");
        }

        veinsmobility: <veinsmobilityType> like
org.car2x.veins.base.modules.IMobility {
            parameters:
                @display("p=130,172;i=block/cogwheel");
        }

    connections:
        nic.upperLayerOut --> appl.lowerLayerIn;
        nic.upperLayerIn <-- appl.lowerLayerOut;
        nic.upperControlOut --> appl.lowerControlIn;
        nic.upperControlIn <-- appl.lowerControlOut;

        veinsradioIn --> nic.radioIn;
}
```

Inter-module Communication

- OMNeT is oriented to **discrete events** and **multi-agent**
- Both capabilities are enabled by **message passing** – either by passing **messages to other nodes**, either by **scheduling messages to the future**
- The **handleMessage()** is a standard method in OMNeT that allows a module to receive any kind of messages.
  - ALL MODULES THAT SEND/RECEIVE MESSAGES HAVE IT
- To send a message, you can use:
  - send() family of functions - to send messages to other modules
  - scheduleAt() - to schedule an event (module sends a message to itself)



projects - veins/src/veins/modules/application/ieee80211p/DemoBaseApp

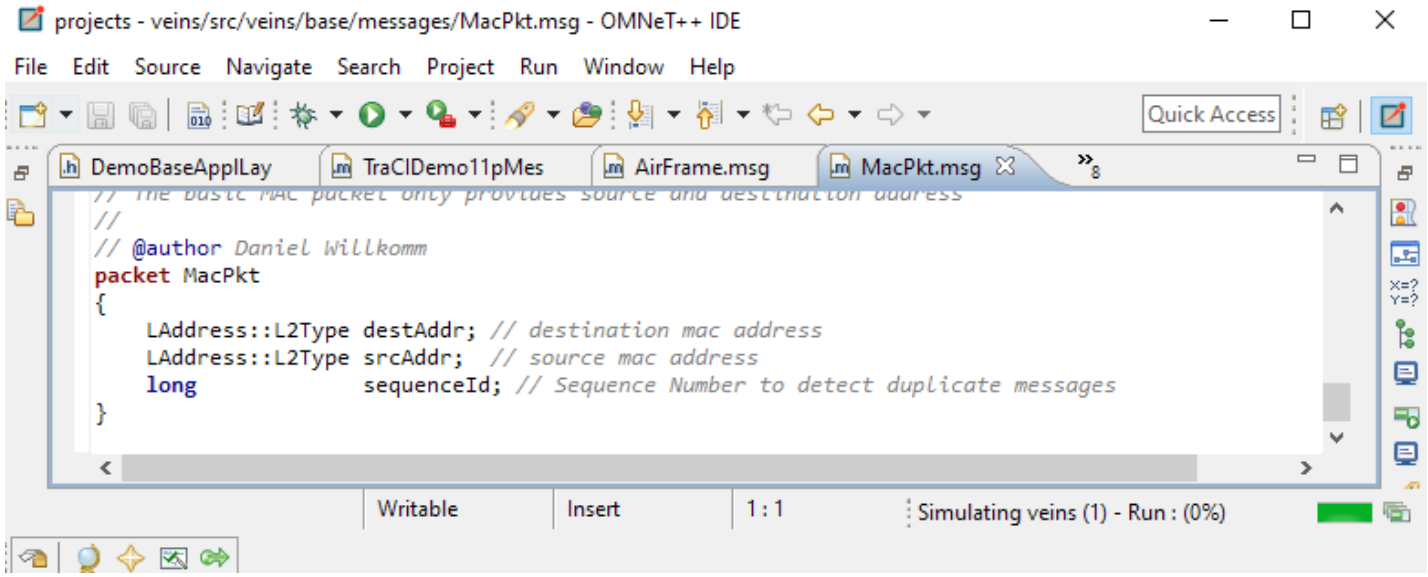
File Edit Source Refactor Navigate Search Project Run Window

Scenario.ned | DemoBaseApplLay | DemoBaseApplLay

```
class VEINS_API DemoBaseApplLayer : public BaseAppl

public:
    ~DemoBaseApplLayer() override;
    void initialize(int stage) override;
    void finish() override;

    void receiveSignal(cComponent* source, simsigna

    enum DemoApplMessageKinds {
        SEND_BEACON_EVT,
        SEND_WSA_EVT
    };

protected:
    /** @brief handle messages from below and calls
    void handleLowerMsg(cMessage* msg) override;

    /** @brief handle self messages */
    void handleSelfMsg(cMessage* msg) override;
```

The MSG syntax

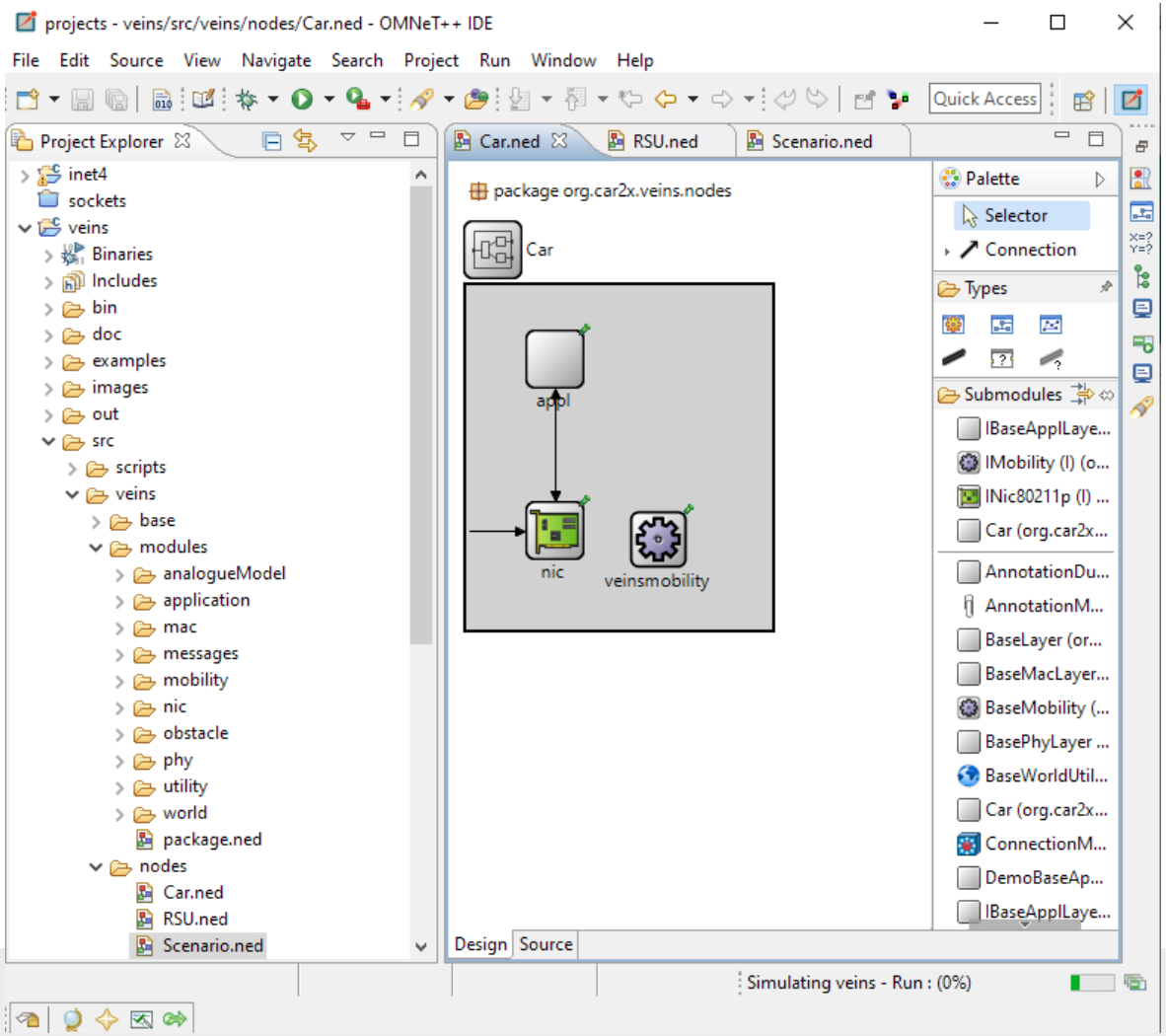
- There is also a file format and syntax to defined messages - **.msg**
- Fairly simpler than NED syntax, as messages are mostly a list of fields
- At pre-compile time, a class (.cc & .h files) are created automatically – very useful stuff!!!



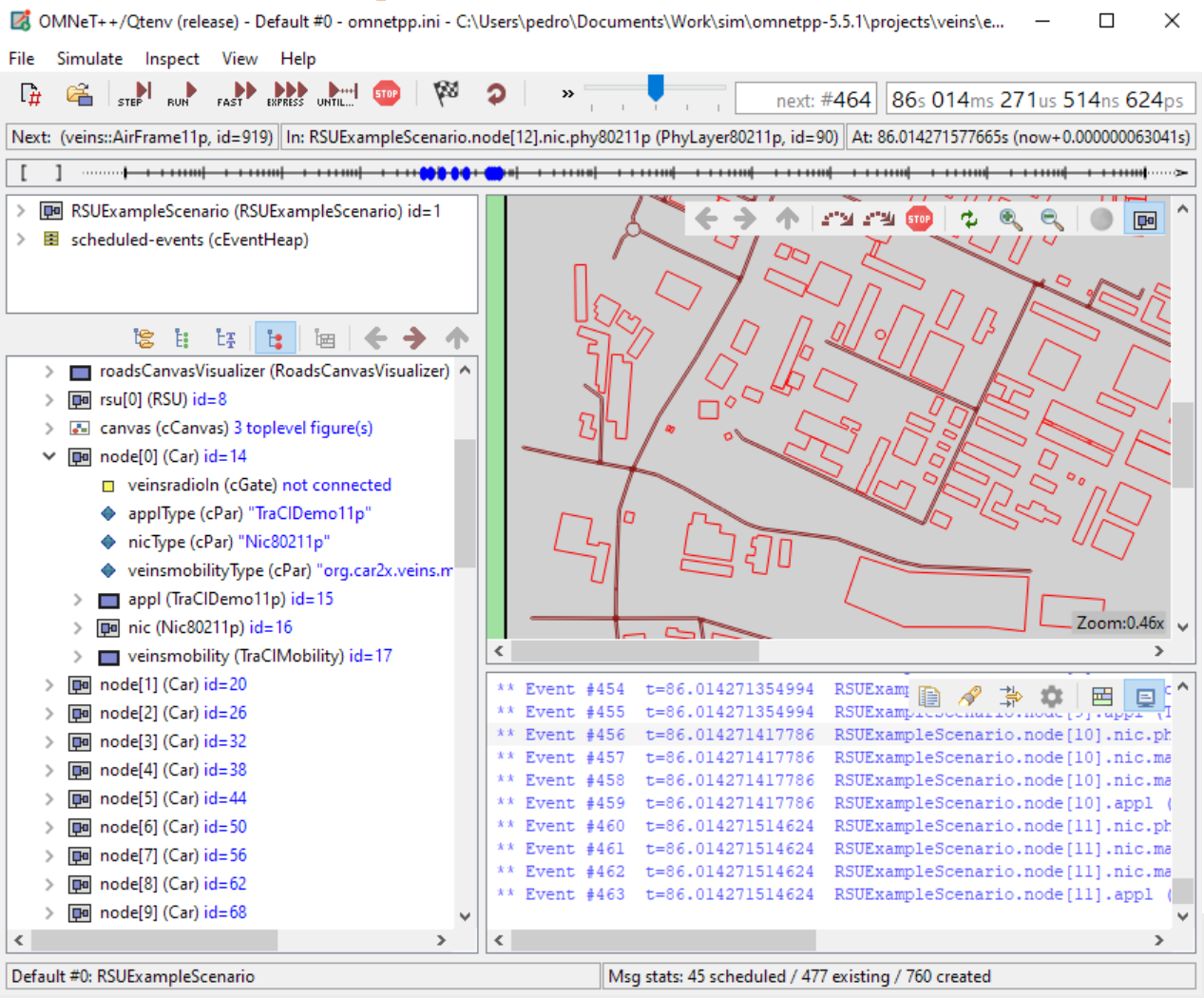
A bit more on modules

Graphical Interfaces

IDE (Integrated Development Environment; Eclipse-based)



Graphical Interface (Qt-based)



Inspection of Module Composition

IDE (Integrated Development Environment; Eclipse-based)

projects - veins/src/veins/nodes/Car.ned - OMNeT++ IDE

File Edit Source View Navigate Search Project Run Window Help

Project Explorer

- inet4
- sockets
- veins
  - Binaries
  - Includes
  - bin
  - doc
  - examples
  - images
  - out
  - src
    - scripts
    - veins
      - base
      - modules
        - analogueModel
        - application
        - mac
        - messages
        - mobility
        - nic
        - obstacle
        - phy
        - utility
        - world
        - package.ned
      - nodes
        - Car.ned
        - RSU.ned
        - Scenario.ned

Car.ned | RSU.ned | Scenario.ned

package org.car2x.veins.nodes

Car

appl

nic

veinsmobility

Palette: Selector, Connection, Types, Submodules

IBaseApplLaye... IMobility (I) (o... INic80211p (I) ... Car (org.car2x... AnnotationDu... AnnotationM... BaseLayer (or... BaseMacLayer... BaseMobility (... BasePhyLayer ... BaseWorldUtil... Car (org.car2x... ConnectionM... DemoBaseAp... IBaseApplLaye...

Design | Source

Simulating veins - Run : (0%)

Graphical Interface (Qt-based)

OMNeT++/Qtenv (release) - Default #0 - omnetpp.ini - C:\Users\pedro\Documents\Work\sim\omnetpp-5.5.1\projects\veins\e...

File Simulate Inspect View Help

STEP RUN FAST EXPRESS UNTIL... STOP

next: #464 | 86s 014ms 271us 514ns 624ps

Next: (veins::AirFrame11p, id=919) | In: RSUExampleScenario.node[12].nic.phy80211p (PhyLayer80211p, id=90) | At: 86.014271577665s (now+0.000000063041s)

- RSUExampleScenario (RSUExampleScenario) id=1
- scheduled-events (cEventHeap)
- roadsCanvasVisualizer (RoadsCanvasVisualizer)
- rsu[0] (RSU) id=8
- canvas (cCanvas) 3 toplevel figure(s)
- node[0] (Car) id=14
  - veinsradioIn (cGate) not connected
  - applType (cPar) "TraCIDemo11p"
  - nicType (cPar) "Nic80211p"
  - veinsmobilityType (cPar) "org.car2x.veins.m
  - appl (TraCIDemo11p) id=15
  - nic (Nic80211p) id=16
  - veinsmobility (TraCIMobility) id=17
- node[1] (Car) id=20
- node[2] (Car) id=26
- node[3] (Car) id=32
- node[4] (Car) id=38
- node[5] (Car) id=44
- node[6] (Car) id=50
- node[7] (Car) id=56
- node[8] (Car) id=62
- node[9] (Car) id=68

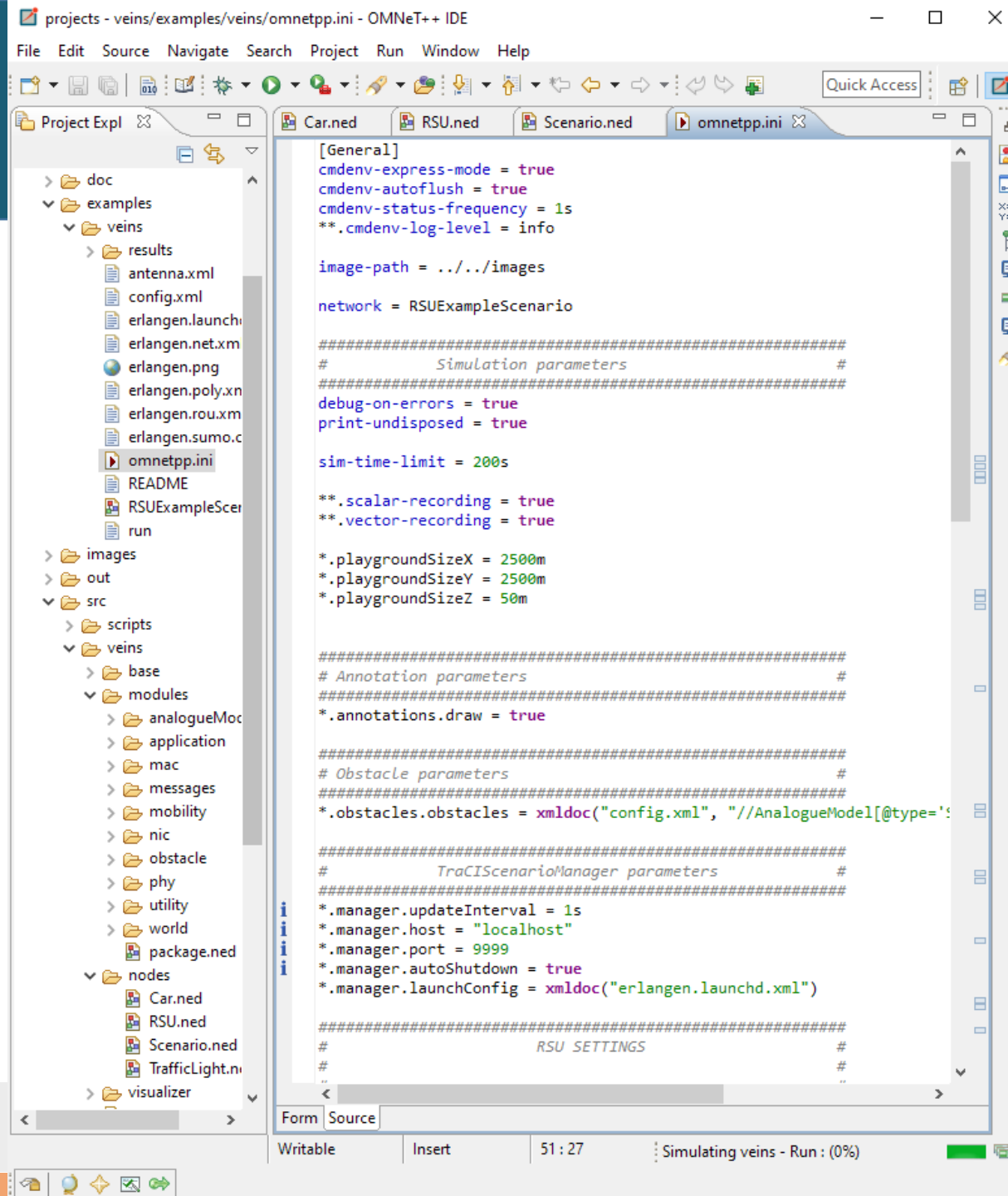
Zoom:0.46x

```
** Event #454  t=86.014271354994  RSUExam
** Event #455  t=86.014271354994  RSUExam
** Event #456  t=86.014271417786  RSUExampleScenario.node[10].nic.ph
** Event #457  t=86.014271417786  RSUExampleScenario.node[10].nic.ma
** Event #458  t=86.014271417786  RSUExampleScenario.node[10].nic.ma
** Event #459  t=86.014271417786  RSUExampleScenario.node[10].appl (
** Event #460  t=86.014271514624  RSUExampleScenario.node[11].nic.ph
** Event #461  t=86.014271514624  RSUExampleScenario.node[11].nic.ma
** Event #462  t=86.014271514624  RSUExampleScenario.node[11].nic.ma
** Event #463  t=86.014271514624  RSUExampleScenario.node[11].appl (
```

Default #0: RSUExampleScenario | Msg stats: 45 scheduled / 477 existing / 760 created

omnetpp.ini

- The omnetpp.ini file contains all the network configurations



Specifying Modules

- Composing modules can be specified at NED file or omnetpp.ini

projects - veins/src/veins/nodes/RSU.ned - OMNeT++ IDE

File Edit Source View Navigate Search Project Run Window Help

Car.ned | RSU.ned | Scenario.ned | omnetpp.ini

```
package org.car2x.veins.nodes;

import org.car2x.veins.base.modules.*;
import org.car2x.veins.modules.nic.Nic80211p;

module RSU
{
    parameters:
        string applType; //type of the application layer
        string nicType = default("Nic80211p"); // type of network interface card
    gates:
        input veinsradioIn; // gate for sendDirect
    submodules:
        appl: <applType> like org.car2x.veins.base.modules.IBaseApplLayer {
            parameters:
                @display("p=60,50");
        }

        nic: <nicType> like org.car2x.veins.modules.nic.INic80211p {
            parameters:
                @display("p=60,166");
        }

        mobility: BaseMobility {
            parameters:
                @display("p=130,172;i=block/cogwheel");
        }

    connections:
        nic.upperLayerOut --> appl.lowerLayerIn;
        nic.upperLayerIn <-- appl.lowerLayerOut;
        nic.upperControlOut --> appl.lowerControlIn;
        nic.upperControlIn <-- appl.lowerControlOut;

        veinsradioIn --> nic.radioIn;
}
```

Design | Source

Writable | Insert | 1 : 1 | Simulating veins - Run : (0%)

projects - veins/examples/veins/omnetpp.ini - OMNeT++ IDE

File Edit Source Navigate Search Project Run Window Help

Car.ned | RSU.ned | Scenario.ned | omnetpp.ini

```
#            TraCIScenarioManager parameters            #
#######################################################
*.manager.updateInterval = 1s
*.manager.host = "localhost"
*.manager.port = 9999
*.manager.autoShutdown = true
*.manager.launchConfig = xmldoc("erlangen.launchd.xml")

#######################################################
#                      RSU SETTINGS                   #
#                                                     #
#                                                     #
#######################################################
*.rsu[0].mobility.x = 2000
*.rsu[0].mobility.y = 2000
*.rsu[0].mobility.z = 3

*.rsu[*].applType = "TraCIDemoRSU11p"
*.rsu[*].appl.sendBeacons = false
*.rsu[*].appl.dataOnSch = false
*.rsu[*].appl.beaconInterval = 1s
*.rsu[*].appl.beaconUserPriority = 7
*.rsu[*].appl.dataUserPriority = 5
*.rsu[*].nic.phy80211p.antennaOffsetZ = 0 m

#######################################################
#            11p specific parameters                  #
#                                                     #
#                    NIC-Settings                     #
#######################################################
*.connectionManager.sendDirect = true
*.connectionManager.maxInterfDist = 2600m
*.connectionManager.drawMaxIntfDist = false

*.**.nic.mac1609_4.useServiceChannel = false
```

Form | Source

Writable | Insert | 51 : 27 | Simulating veins - Run : (0%)

Interfaces

- Sometimes, when preparing a simulation, you don't want to specify a module right
- You can do that through an Interface
- These are typically identified by a capital "I" at the start of the name
  - E.g.: "IWirelessNic"
- You can specify the module to be used in the omnetpp.ini file

projects - veins/src/veins/base/modules/IBaseApplLayer.ned - OMNeT++ IDE

```
package org.car2x.veins.base.modules;

// Interface for application layer modules
moduleinterface IBaseApplLayer
{
    parameters:
        int headerLength @unit("bit"); // length of the application message header (in bits)
    gates:
        input lowerLayerIn; // from network layer
        output lowerLayerOut; // to network layer
        input lowerControlIn; // control from network layer
        output lowerControlOut; // control to network layer
}
```

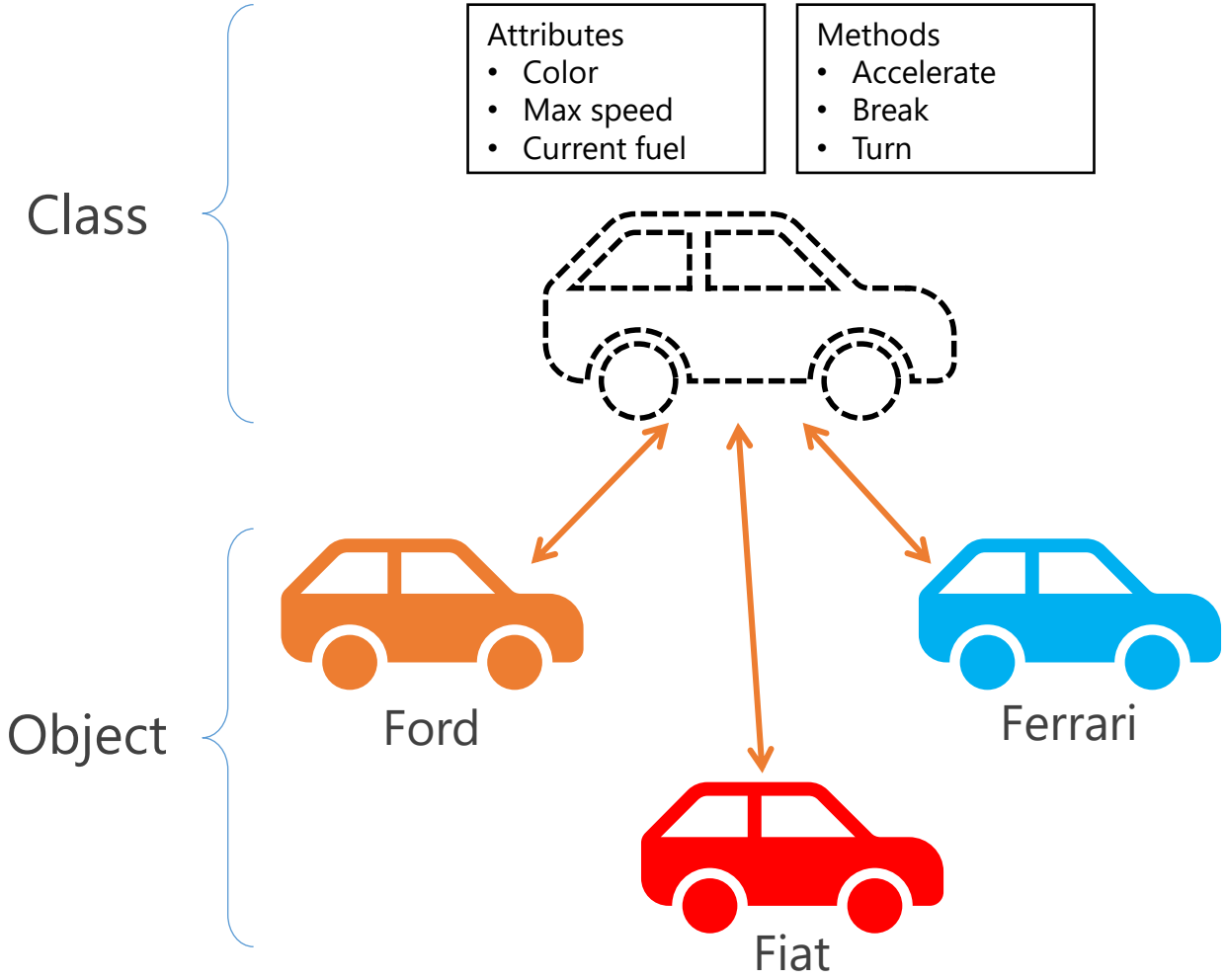
projects - veins/src/veins/modules/application/ieee80211p/DemoBaseApplLayer.ned - OMNeT++ IDE

```
simple DemoBaseApplLayer like IBaseApplLayer
{
    parameters:
        @class(veins::DemoBaseApplLayer);
        int headerLength = default(88bit) @unit(bit); //header length of the application

        bool sendBeacons = default(true); //tell the applayer to periodically send beacons
        int beaconLengthBits = default(256bit) @unit(bit); //the length of a beacon packet
        int beaconUserPriority = default(7); //the user priority (UP) of the beacon messages
        double beaconInterval = default(1s) @unit(s); //the intervall between 2 beacon messages

        int dataLengthBits = default(1024bit) @unit(bit); //the length of a data packet
        bool dataOnSch = default(false); //tells the applayer whether to use a service channel for datapacke
        int dataUserPriority = default(7); //the default user priority (UP) for data packets

        bool avoidBeaconSynchronization = default(true); //don't start beaconing directly after node was cre
```

Object-Oriented Programming

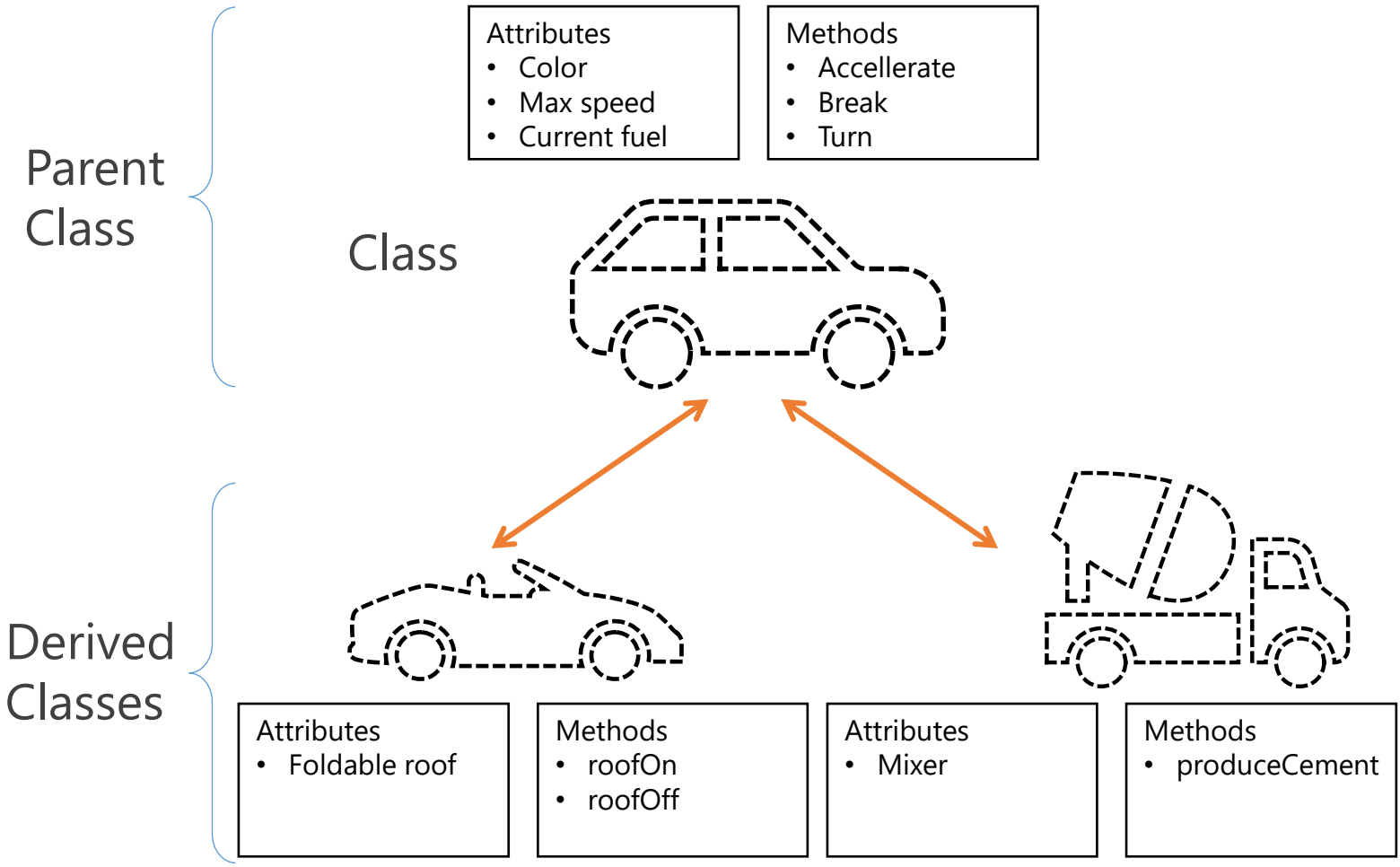
Object-Oriented Programming

- **Objects:** instances of a class, with their own attribute values and similar methods
- **Classes:** a description of the objects in terms of attributes and methods
- Analogy:
  - **Class:** blueprint; corresponds to the source code
  - **Objects:** physical objects; only exist in run-time



Object-Oriented Programming

- Inheritance of methods and data
  - New classes can be **derived** from existing ones
  - Derived classes keep all attributes and methods of parent classes
  - Derived classes can define new attributed/methods, or even override (re-define) existing methods
- Other properties
  - Polymorphism in Base/Derived classes
  - Constructor method
  - Abstraction
  - Encapsulation of internal data



Derived Modules

- Very often modules in OMNeT++ are derived from existing modules
- Derivation can go quite deep; sometimes 5+ derivations
- Parent class(es) functions with same name can be overridden
- Initialization Procedure
  - Initialization of parent classes is (typically) called
  - Modules have 2 initialization stages

projects - veins/src/veins/modules/application/ieee80211p/DemoBaseApplLayer.h - OMNeT++ IDE

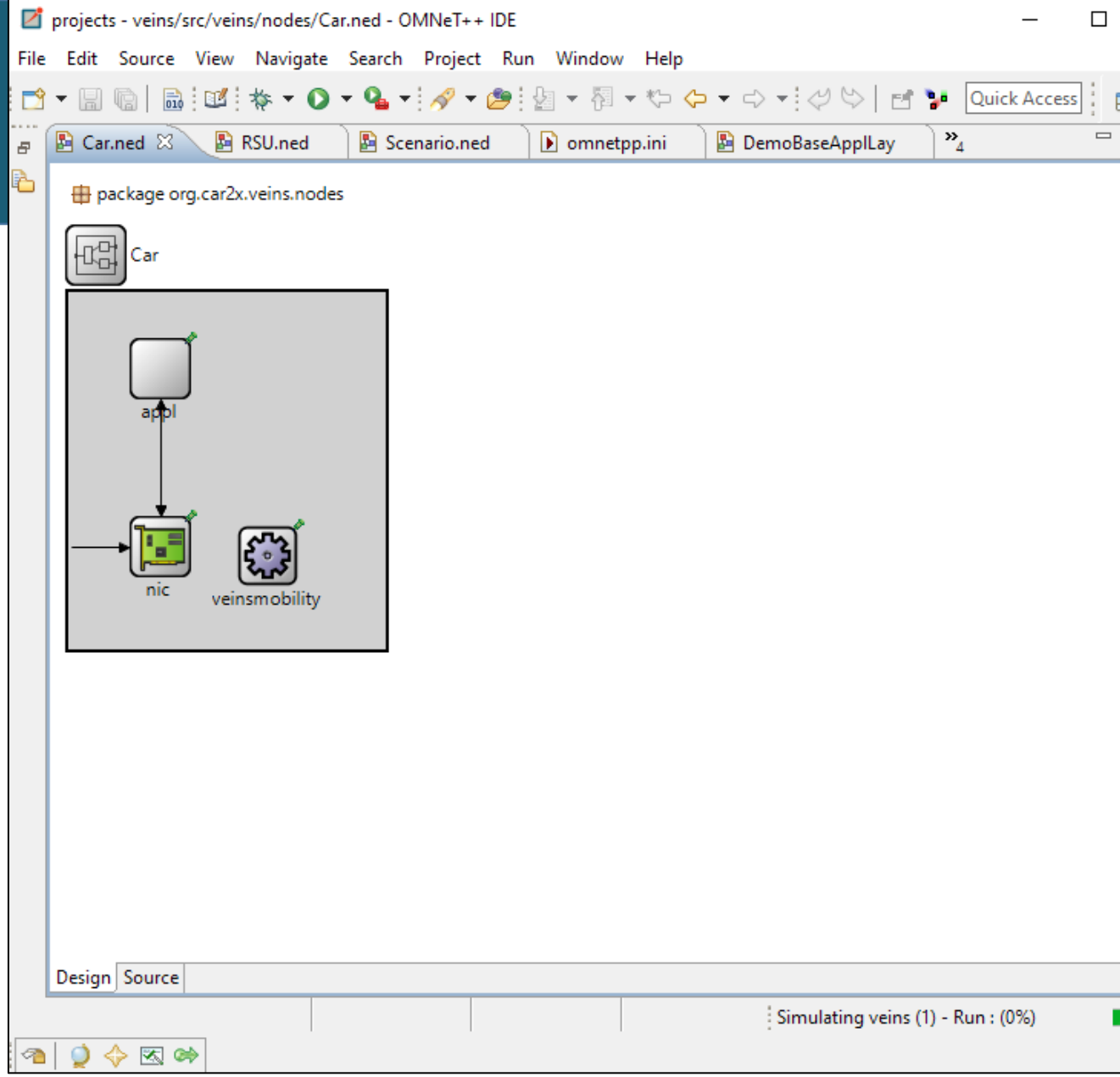
```
class VEINS_API DemoBaseApplLayer : public BaseApplLayer {

public:
    ~DemoBaseApplLayer() override;
    void initialize(int stage) override;
    void finish() override;

    void receiveSignal(cComponent* source, simsignal_t signalID, cObject* obj, cObject* details) override;

    enum DemoApplMessageKinds {
        SEND_BEACON_EVT,
        SEND_WSA_EVT
    };
```

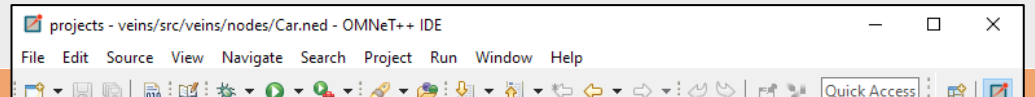
projects - veins/src/veins/modules/application/ieee80211p/DemoBaseApplLayer.cc - OMNeT++ IDE

```
void DemoBaseApplLayer::initialize(int stage)
{
    BaseApplLayer::initialize(stage);

    if (stage == 0) {

        // initialize pointers to other modules
        if (FindModule<TraCIMobility*>::findSubModule(getParentModule())) {
            mobility = TraCIMobilityAccess().get(getParentModule());
            traci = mobility->getCommandInterface();
            traciVehicle = mobility->getVehicleCommandInterface();
        }
        else {
            traci = nullptr;
            mobility = nullptr;
            traciVehicle = nullptr;
        }
```

A Compound Module

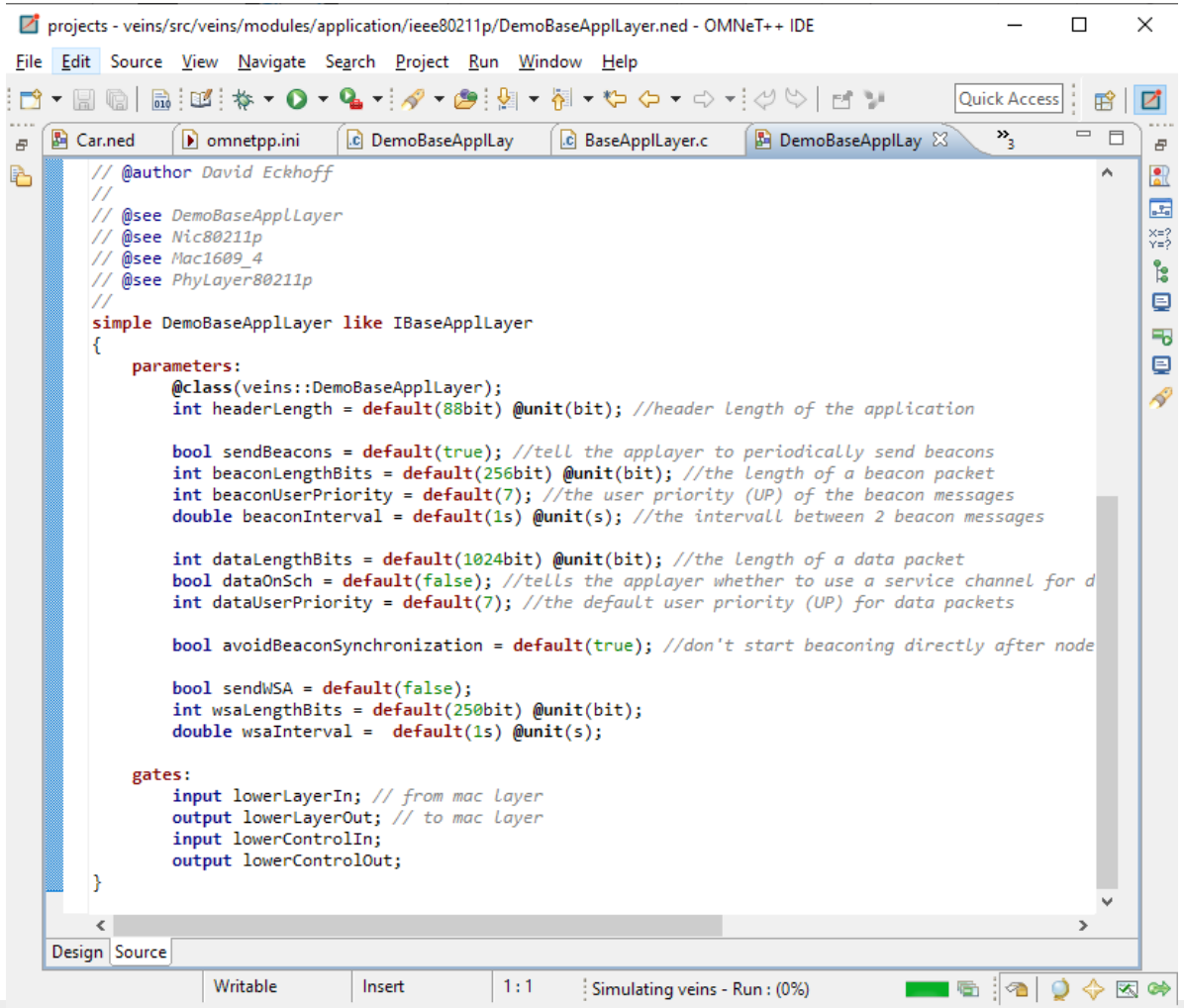
```
module Car
{
    parameters:
        string applType; //type of the application layer
        string nicType = default("Nic80211p"); // type of network interface card
        string veinsmobilityType =
default("org.car2x.veins.modules.mobility.traci.TraCIMobility"); //type of the
mobility module
    gates:
        input veinsradioIn; // gate for sendDirect
    submodules:
        appl: <applType> like org.car2x.veins.base.modules.IBaseApplLayer {
            parameters:
                @display("p=60,50");
        }

        nic: <nicType> like org.car2x.veins.modules.nic.INic80211p {
            parameters:
                @display("p=60,166");
        }

        veinsmobility: <veinsmobilityType> like org.car2x.veins.base.modules.IMobility
{
            parameters:
                @display("p=130,172;i=block/cogwheel");
        }

    connections:
        nic.upperLayerOut --> appl.lowerLayerIn;
        nic.upperLayerIn <-- appl.lowerLayerOut;
        nic.upperControlOut --> appl.lowerControlIn;
        nic.upperControlIn <-- appl.lowerControlOut;

        veinsradioIn --> nic.radioIn;
}
```





IMPORTANT: COMPOSITION is not DERIVATION

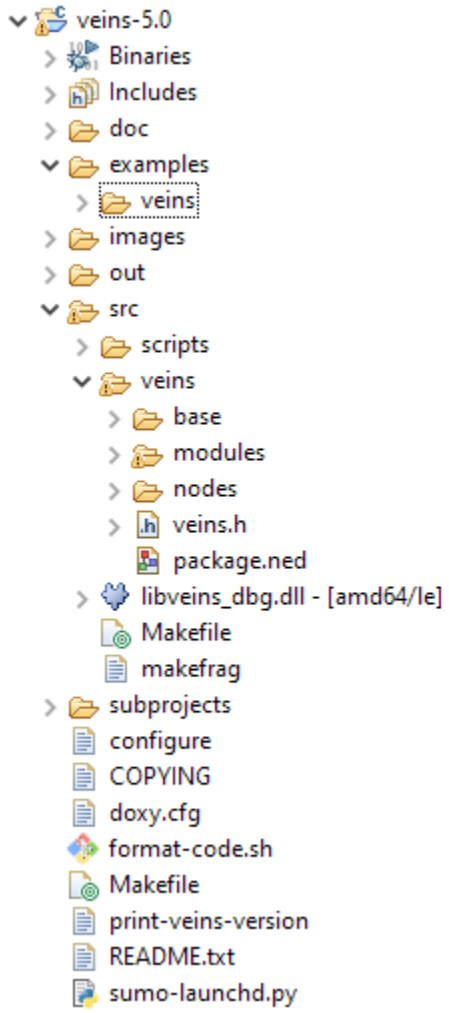
- Note: these are two concepts are diferent:
- **Module Composition / Nesting:**
  - only applies in NED syntax, in which compound modules can have nested modules
  - There's also nesting in C++, but it is not relevant in this case
- **Derivation:**
  - Something specific to C++ (and other languages)
  - Helpful to re-use and extend existing code
  - Should not be confused with NED module composition/nesting
  - I will find no reference to derivation in NED files



Practical Aspects

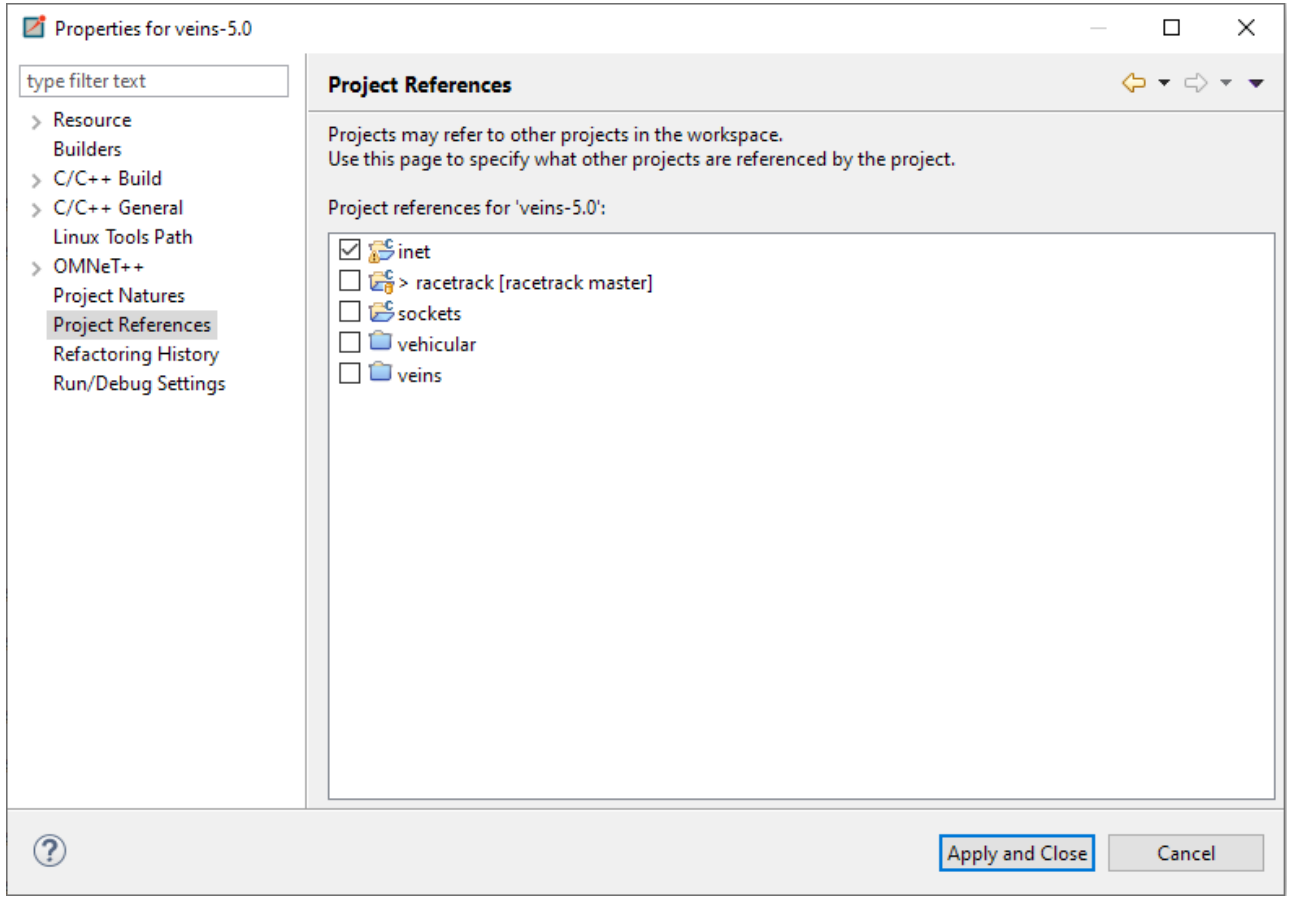
Typical Project Structure

- src: contains all source code. Typically divided into:
  - **base:** basic classes/modules for all components of the library
  - **modules:** evolved modules, typically derived from base modules
  - **nodes:** typically agents or network nodes



Project Referencing

- When using projects that reference other projects, the projects need to reference each other




isep
CISTER - Research Center in
Real-Time & Embedded Computing Systems
New Frontiers
in Computing

Thank You