Tutorial 5: Basic Platooning Implementation

Basic Platooning Implementation

Prof. Sangyoung Park

Module ”Vehicle-2-X: Communication and Control“

Implementation of Simple Platooning Algorithm

- Let‘s start with something simple
- Let‘s read the distance to the preceding vehicle only and try to adjust the acceleration of the current vehicle
- Would you be able to implement this?
- $a = p \cdot (d - d_{desired})$

Letting all nodes to broadcast

- Previously, we were allowing only the lead vehicle to broadcast
- For now, let‘s allow all the vehicles to broadcast
- In initilize(),
  - Only the lead vehicle announces the service (but tbh, it‘s not necessary)
  - Move the scheduleAt() out of the if clause
  - Notice the change of ID from 14 to 13? Let me explain this in the next slides

```
else if (stage == 1) {
        //Initializing members that require initialized other modules goes here
        // Vehicle IDs are 14, 20, 26, 32, and 38, respectively

        if (myId == 13){ // this is the head vehicle
            startService(Channels::SCH2, currentOfferedServiceId, "Platoon Lead Vehicle Service");
        }
        scheduleAt(computeAsynchronousSendingTime(beaconInterval, type_CCH),sendBeaconEvt);
    }
```

Identifying the Sender

- Now, we can’t assume the sender is the lead vehicle
- We need to identify the sender upon receiving BSM before taking action
- Read the sender ID from BSM and check whether it’s from the preceding vehicle
- Let’s define a Boolean variable to do so (from PrecedingVehicle)

```
void VehicleControlApp::onBSM(BasicSafetyMessage* bsm){
    int senderId = bsm->getSenderAddress();

    bool fromPrecedingVehicle = false;

    switch (this->myId){
        case 13:
            break;
        case 19:
            if (senderId == 13 ) fromPrecedingVehicle = true;
            break;
        case 25:
            if (senderId == 19 ) fromPrecedingVehicle = true;
            break;
        case 31:
            if (senderId == 25 ) fromPrecedingVehicle = true;
            break;
        case 37:
            if (senderId == 31 ) fromPrecedingVehicle = true;
            break;
        default:
            ASSERT(0); // no other ids should exist in the simulation
            break;
    } .....
```

Identifying the Sender

- I thought that this->getId() would yield a unique identifier, but it seems that the return value of getId() is a component in the lower layer
- The ID of the VehicleControlApp can be obtained by directly accessing myId of the class
- This is also equivalent to the senderId populated by populateWSM()
- So, the IDs will be now 13, 19, 25, 31, 37, ...

Adjust the Vehicle Velocity

- In onBSM(), we could adjust the acceleration (or speed) to change the status of the vehicle if fromPrecedingVehicle == true

```
    Coord& precedingVehicleSpeed = bsm->getSenderSpeed();
    Coord& precedingVehiclePos = bsm->getSenderPos();
    traciVehicle->setSpeedMode(0x1f);

    double desiredDistance = 6.0;
    double coeff = 1;

    if (fromPrecedingVehicle)
    {
        double distance = (precedingVehiclePos- curPosition).length();
        double acc = coeff * (distance - desiredDistance);

        std::cout << "t" << simTime() << ": Distance [" << senderId << "]-[" << myId << "]: " << distance <<
" acc: " << acc << "\n";

        if ( acc > 0){
            traciVehicle->setAccel(acc);
            traciVehicle->setSpeed(100);
        }
        else {
            traciVehicle->setDecel(-acc);
            traciVehicle->setSpeed(0);
        }
    }
```

Setting Acceleration Values

- Veins does not provide an interface to directly control the acceleration of vehicles
- We could do the following work around (maybe there's a better way)
  - Set maximum acceleration or deceleration value
  - Set a very high speed or low speed to ensure that the vehicle is taking that maximum acceleration or deceleration value
  - I am open to suggestions or improvements
- But Veins doesn't provide an interface to control the *max acceleration* and *max deceleration* either
- Let's try implement the functionalities

New Functions to the TraCI Interface

- TraCI interface is no magic, all the commands and API (functions we could use) are defined in the following three files in the folder veins/src/veins/modules/mobility/traci/
  - TraCICommandInterface.cc and TraCICommandInterface.h
  - TraCIConstants.h
- For example, if you look at the function we already used, “setSpeed()”
  - You can see that variableId = VAR_SPEED
  - VAR_SPEED is defined in TraCIConstants.h as 0x40
  - You can also see 0x40 from https://sumo.dlr.de/wiki/TraCI/Change_Vehicle_State

```
void TraCICommandInterface::Vehicle::setSpeed(double speed) {
uint8_t variableId = VAR_SPEED;
uint8_t variableType = TYPE_DOUBLE;
TraCIBuffer buf = traci->connection.query(CMD_SET_VEHICLE_VARIABLE, TraCIBuffer() << variableId << nodeId << 
variableType << speed);
ASSERT(buf.eof());
}
```

New Functions to the TraCI Interface

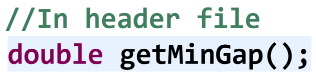
- So, we could implement the functions setAccel() and setDecel() in a similar way
- Define the function format in the header file (.h)
- Define the function in the cc file (.cc)

```
// in TraCICommandInterface.h
void setAccel(double accel);
void setDecel(double decel);
```

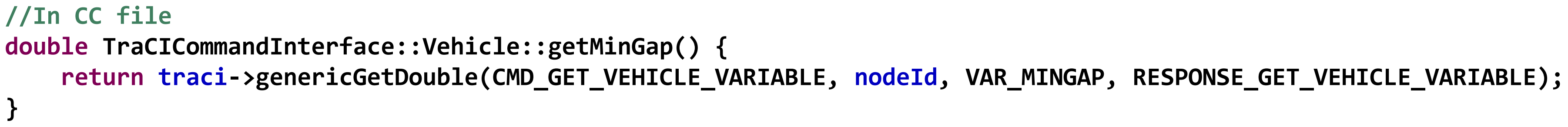
```
// added by spark
void TraCICommandInterface::Vehicle::setAccel(double
accel) {
    uint8_t variableId = VAR_ACCEL;
    uint8_t variableType = TYPE_DOUBLE;
    TraCIBuffer buf = traci-
>connection.query(CMD_SET_VEHICLE_VARIABLE, TraCIBuffer()
<< variableId << nodeId << variableType << accel);
    ASSERT(buf.eof());
}

// added by spark
void TraCICommandInterface::Vehicle::setDecel(double
decel) {
    uint8_t variableId = VAR_DECEL;
    uint8_t variableType = TYPE_DOUBLE;
    TraCIBuffer buf = traci-
>connection.query(CMD_SET_VEHICLE_VARIABLE, TraCIBuffer()
<< variableId << nodeId << variableType << decel);
    ASSERT(buf.eof());
}
```

What About Reading Variables using TraCI?

- For example, you can read the minGap parameter in the car following model (recall car following model lecture)
- https://sumo.dlr.de/wiki/Definition_of_Vehicles,_Vehicle_Types,_and_Routes



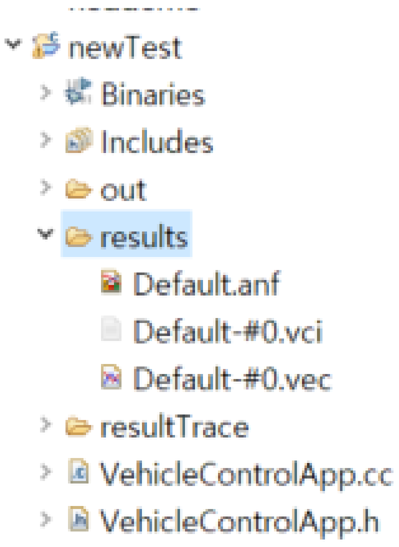
```
//In header file
double getMinGap();
```



```
//In CC file
double TraCICommandInterface::Vehicle::getMinGap() {
    return traci->genericGetDouble(CMD_GET_VEHICLE_VARIABLE, nodeId, VAR_MINGAP, RESPONSE_GET_VEHICLE_VARIABLE);
}
```

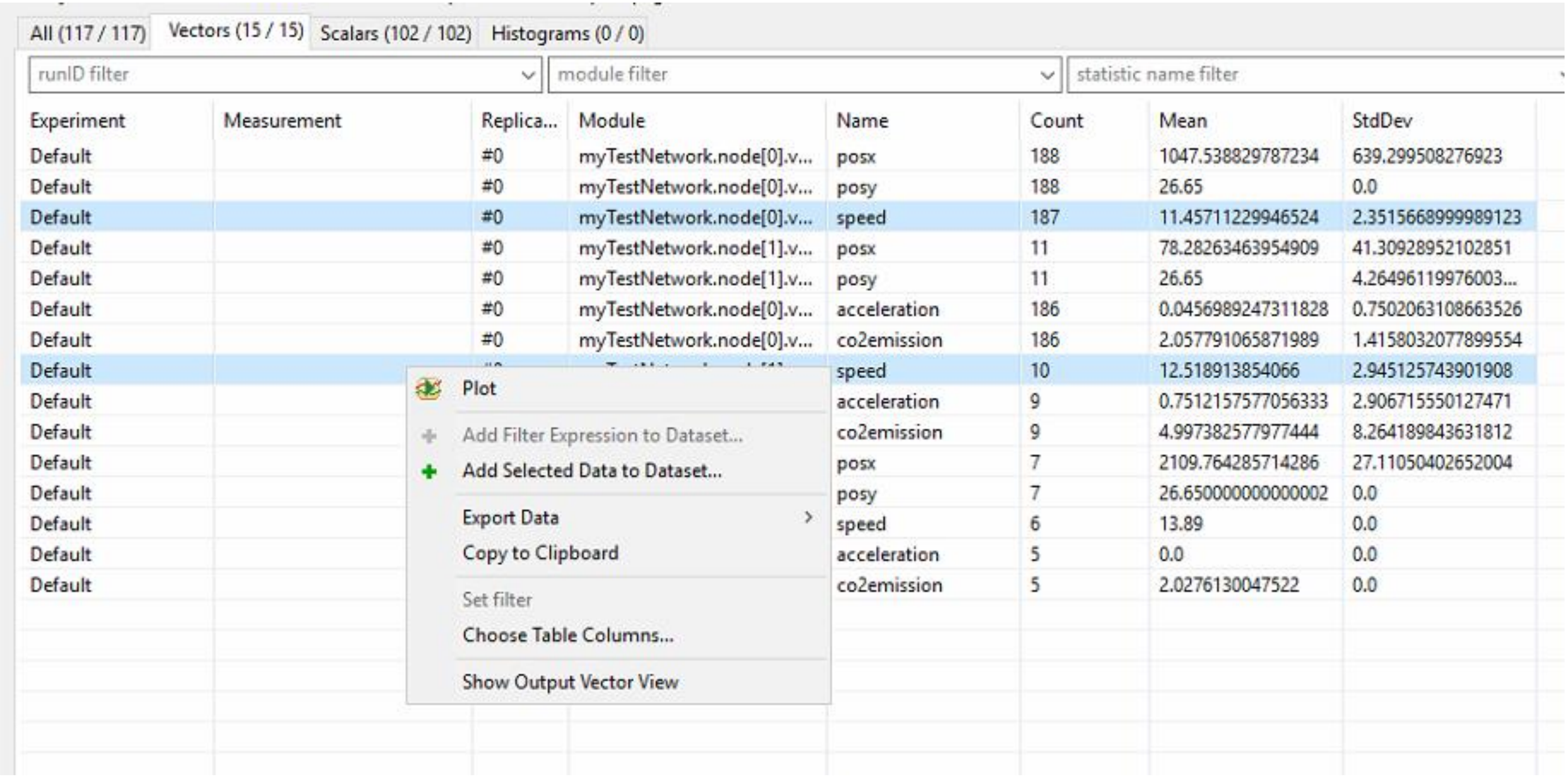
Plotting the Results

- Fortunately, Veins provides its own statistics mechanism, so we can just make use of it
- After you simulate anything, data will be generated in the results folder
- If you double click .vec file you will be able to generate .anf file
- In the tab “browse data” at the bottom,
  and then “vectors” tab at the top,
  you will be able to generate graphs about
  the position, velocity, and acceleration of vehicles



```
newTest
  Binaries
  Includes
  out
  results
    Default.anf
    Default-#0.vci
    Default-#0.vec
  resultTrace
  VehicleControlApp.cc
  VehicleControlApp.h
```

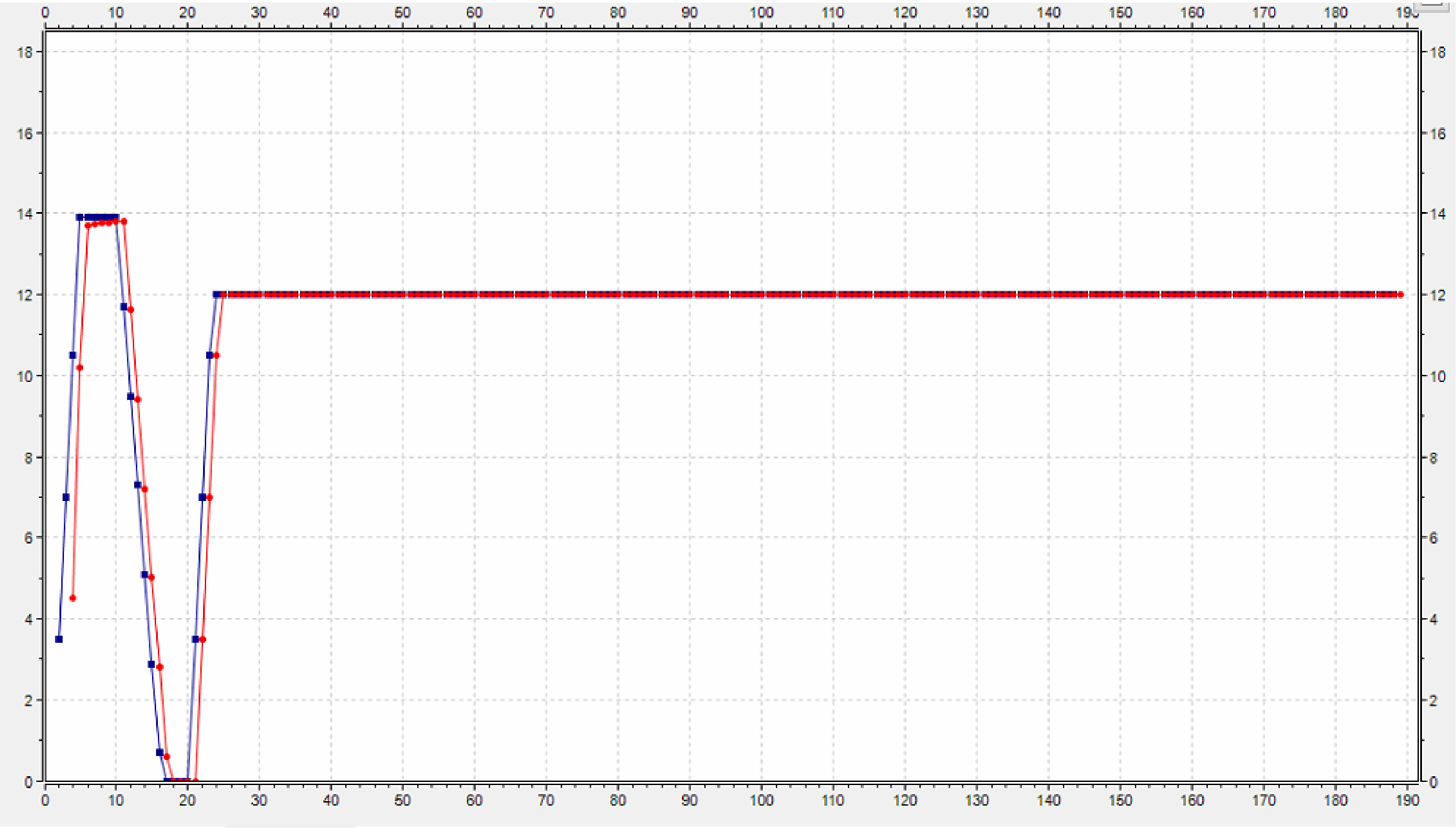
Plotting the Results



Experiment	Measurement	Replica...	Module	Name	Count	Mean	StdDev
Default		#0	myTestNetwork.node[0].v...	posx	188	1047.538829787234	639.299508276923
Default		#0	myTestNetwork.node[0].v...	posy	188	26.65	0.0
Default		#0	myTestNetwork.node[0].v...	speed	187	11.45711229946524	2.351566899989123
Default		#0	myTestNetwork.node[1].v...	posx	11	78.28263463954909	41.30928952102851
Default		#0	myTestNetwork.node[1].v...	posy	11	26.65	4.26496119976003...
Default		#0	myTestNetwork.node[0].v...	acceleration	186	0.0456989247311828	0.7502063108663526
Default		#0	myTestNetwork.node[0].v...	co2emission	186	2.057791065871989	1.415803207789554
Default				speed	10	12.518913854066	2.945125743901908
Default				acceleration	9	0.7512157577056333	2.906715550127471
Default				co2emission	9	4.997382577977444	8.264189843631812
Default				posx	7	2109.764285714286	27.11050402652004
Default				posy	7	26.650000000000002	0.0
Default				speed	6	13.89	0.0
Default				acceleration	5	0.0	0.0
Default				co2emission	5	2.0276130047522	0.0

Speed vs Time Graph

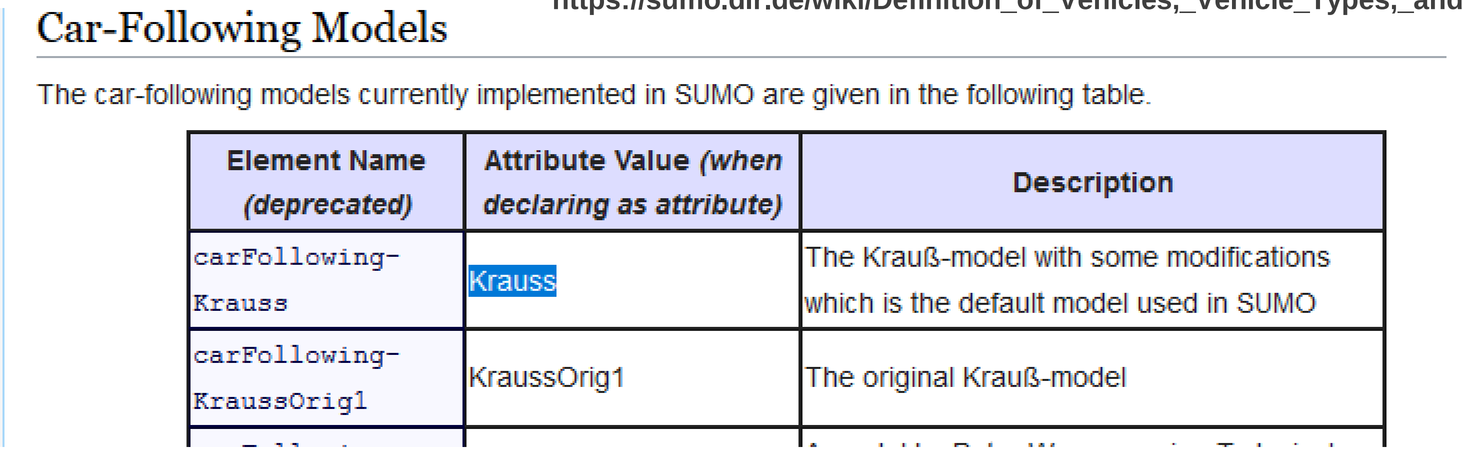
- Wait why is the the velocity the same and the gap is 19.05 m? The control doesn’t work!



Overriding the SUMO Driver Models

- One thing to note is that SUMO does not allow direct control of vehicle acceleration and deceleration, but rather lets you configure parameters in “driver models”
- SUMO default model is “carFollowing-Krauss”

https://sumo.dlr.de/wiki/Definition_of_Vehicles,_Vehicle_Types,_and_Routes



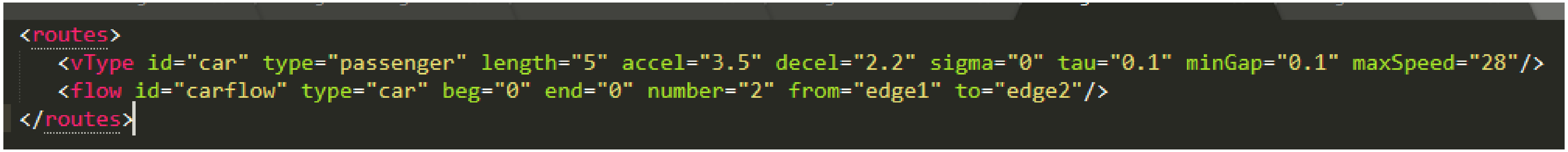
Car-Following Models

The car-following models currently implemented in SUMO are given in the following table.

Element Name ***(deprecated)***	**Attribute Value** ***(when declaring as attribute)***	**Description**
`carFollowing-Krauss`	Krauss	The Krauß-model with some modifications which is the default model used in SUMO
`carFollowing-KraussOrig1`	KraussOrig1	The original Krauß-model

Overriding the SUMO Driver Models

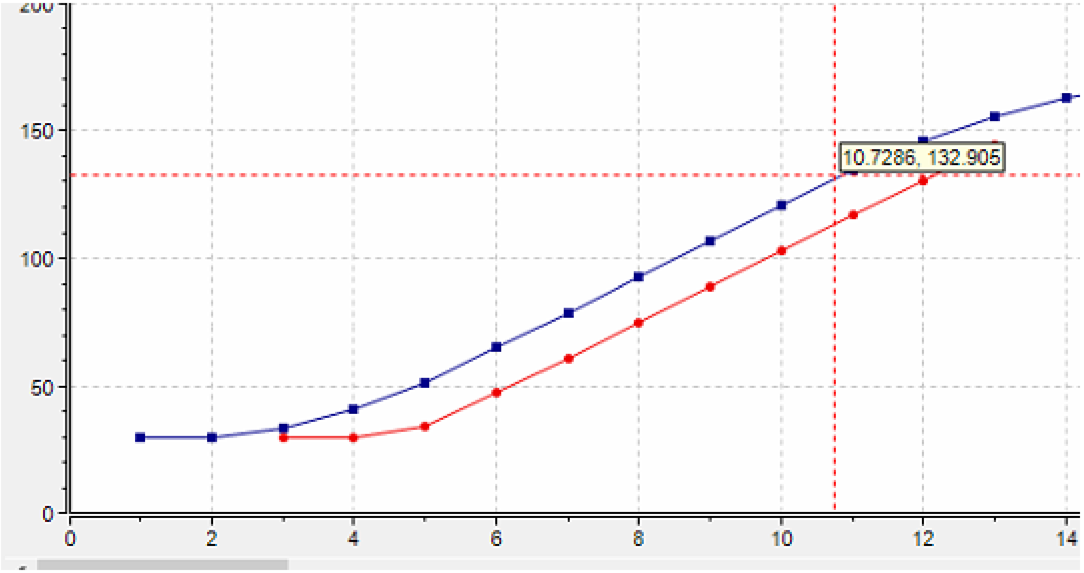
- So, the vehicles are trying to maintain the minimum time and space gap to the preceding vehicle
- The distance we'd like to achieve 6 m is going to be overridden by the driver model from SUMO
- So far, I haven't found a way to directly control acceleration, but we can try to do it by setting the values minGap (space headway) and tau (time headway) to a small value
- We can do that in the .rou.xml file
- Let's say we set the values tau and minGap to be both 0.1 (default values are 1.0 and 2.5)



```
<routes>
    <vType id="car" type="passenger" length="5" accel="3.5" decel="2.2" sigma="0" tau="0.1" minGap="0.1" maxSpeed="28"/>
    <flow id="carflow" type="car" beg="0" end="0" number="2" from="edge1" to="edge2"/>
</routes>
```

Debugging the Code

- Something has happened
- Vehicles collide and disappear in the simulation because our algorithm can’t handle the situation
- X pos vs time graph
  - Red line disappears after 13 seconds



Debugging the Code

- Something’s not right about the results, the positions are not being updated frequently as we want
- If you look into the console window (in Omnetpp IDE), something is wrong
- The vehicle distance is not as often updated (1 sec interval) as the BSM send interval
- This means we can’t rely on current handleUpdatePosition() to update the position of velocity values of the vehicles

```
t3.029858499977: Distance [13]-[19]: 10.5 acc: 4.5
t3.129870016741: Distance [13]-[19]: 10.5 acc: 4.5
t3.229870016741: Distance [13]-[19]: 10.5 acc: 4.5
t3.329870016741: Distance [13]-[19]: 10.5 acc: 4.5
t3.429870016741: Distance [13]-[19]: 10.5 acc: 4.5
t3.529870016741: Distance [13]-[19]: 10.5 acc: 4.5
t3.629870016741: Distance [13]-[19]: 10.5 acc: 4.5
t3.729870016741: Distance [13]-[19]: 10.5 acc: 4.5
t3.829870016741: Distance [13]-[19]: 10.5 acc: 4.5
t3.929870016741: Distance [13]-[19]: 10.5 acc: 4.5
t4.029870056769: Distance [13]-[19]: 16.5 acc: 10.5
t4.129870056769: Distance [13]-[19]: 16.5 acc: 10.5
```

Debugging the Code

- Let me figure something out about this…. Very soon..

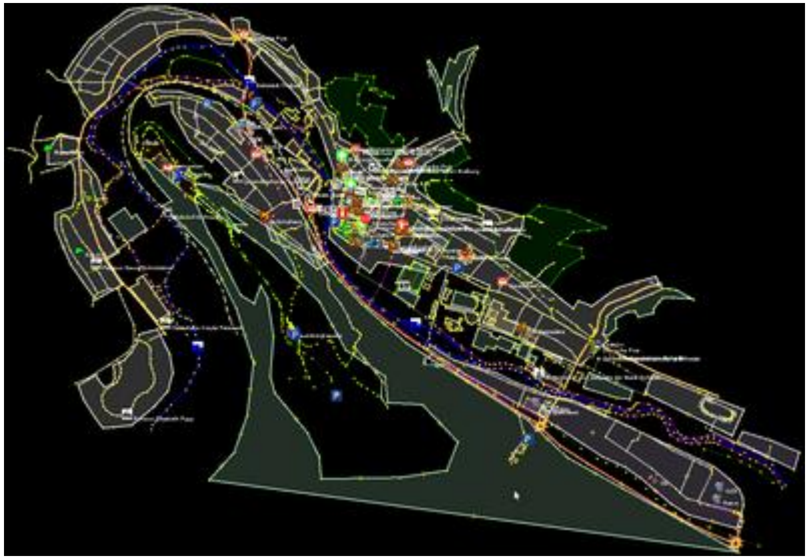
Traffic Light Control

- TraCI interface to traffic light control is given in TraCICommandInterface.cc as well

```
class Trafficlight {
public:
Trafficlight(TraCICommandInterface* traci, std::string trafficLightId) : traci(traci), trafficLightId(trafficLightId)
{
connection = &traci->connection;
}

std::string getCurrentState() const;
int32_t getDefaultCurrentPhaseDuration() const;
std::list<std::string> getControlledLanes() const;
std::list<std::list<TraCITrafficLightLink> > getControlledLinks() const;
int32_t getCurrentPhaseIndex() const;
std::string getCurrentProgramID() const;
TraCITrafficLightProgram getProgramDefinition() const;
int32_t getAssumedNextSwitchTime() const;

void setProgram(std::string program);/**< set/switch to different program */
void setPhaseIndex(int32_t index); /**< set/switch to different phase within the program  */
void setState(std::string state);
void setPhaseDuration(int32_t duration); /**< set remaining duration of current phase in milliseconds */
void setProgramDefinition(TraCITrafficLightProgram::Logic program, int32_t programNr);

protected:
TraCICommandInterface* traci;
TraCIConnection* connection;
std::string trafficLightId;
};
```

Importing Realistic Maps

- If you want to work on realistic maps, you can import maps from openstreetmap
- https://sumo.dlr.de/wiki/Tutorials/Import_from_OpenStreetMap



Further Information

- Platooning Extension (PLEXE) available if you want to use it, you can of course use it

Notice for the Term Project

- Notice for the Term Project

Forming the Groups

- 2 students per group

Examples

- I do not expect something too complicated from you
- The project should be doable within the given time frame
- Should resolve your genuine curiosity about V2X communication
    - Platooning algorithm parameter studies
        - Implement some of the existing platooning algorithms
    - Impact of the communication networks
        - When is platooning impossible? How dense should the traffic be?
    - How do we negotiate between sparsely populated vehicles to form a platoon?
        - How do you decide the lead vehicle and size of the platoon in a distributed system like car platoons?
    - What happens if there's a hostile (or compromised) vehicle system within a platoon
        - Can you detect them?
    - How do we distinguish communication packets when there are multiple platoons in vicinity?
    - Could traffic lights be synchronized with platoon lengths to avoid cutting the platoon in half using communication?

Topic Suggestion

- Please form a group and suggest a topic before next Tuesday by email
  - sangyoung.park@tu-berlin.de
- I will be available every tutorial session for help with the programming
- Topics can be very flexible
  - But don‘t choose a topic, which is too complex and require too much manpower unless you are really into it