Platoon Attack Simulation

**1 Overview**

The learning objective of this lab is for students to get familiar with the concept in platoon attack. In this lab, students can learn how to conduct experiments with PLEXE simulator[1] and how to analyze simulation results. After finishing this lab, students will be able to gain a first-hand experience on platoon attack and plexe simulation. Moreover, students will be able to design their own platooning attack experiments.

**2 Lab Environment**

Please refer to *Plexe Version 1.1 Documentation* to install plexe and deploy experiment environment.

For more information, you can access <http://plexe.car2x.org/>. The current latest version is plexe 2.0, but plexe 1.1 is enough for this lab.

You should follow *Plexe Version 1.1 Documentation* to build SUMO and VEINS. Then you need to install OMNeT++ and R to conduct the simulation. After installing all the needed tools, you can follow the documentation to run the example experiments. It can helps you to understand the plexe simulator. At last, before you go to the next step of this lab, you should read through the documentation and understand it as much as you can.

**3 Lab Tasks**

**3.1 Task 1: Set CACC Constant Space**

To set a constant space for CACC, we need to modify the SimplePlatooningApp.cc in plexe-veins. The file path is /src/plexe-veins/src/modules/application/platooning/apps/ SimplePlatooningApp.cc. In the initialize() function, plexe initialize the parameters for platooning. Therefore, at the end of this function, we need to initialize the gap for CACC.

 **Figure 1**

CommandSetCACCConstantSpacing() is the function that sets a fixed gap for platooning. GetExternalId() returns the current vehicle’s id and 5 means 5 meters gap for CACC. You can change the value and conduct the example experiment again to see the difference.

**3.2 Task 2: Set Leader Speed**

To set leader speed for CACC, we need to modify the omnetpp.ini in plexe-veins. The file path is /src/plexe-veins/examples/sinPlatoon/omnetpp.ini. sinPlatoon is an experiment example which shows how platoon is running. Omnetpp.ini is a configuration file, including simulation parameters and mobility parameters. In this file we can set some parameters for the simulation, for example, leader vehicle speed.

 **Figure 2**

The default setting for leader speed is 100 km/h, you can change the value and conduct the example experiment again to see the difference. The following vehicles in the platoon will adjust their speed to follow leader vehicle.

**3.3 Task 3: Leader Crash Attack**

In the leader crash attack, the leader car stops suddenly (intentionally or not) and causes the following cars to crash over each other. To make the leader vehicle stop, we need to set its acceleration to make it decelerate abruptly. This can be done in MSCFModel\_CC.cpp. The file path is /src/plexe-sumo/src/microsim/cfmodels/MSCFModel\_CC.cpp.

 **Figure 3**

\_v() function updates platoon’s movement and set desired acceleration for each car in next step. Therefore, in this function, we add some codes to launch leader crash attack. We use setFixedAcceleration() method to set the acceleration for leader vehicle so that it can stop. The first parameter represents vehicle in platoon. The second parameter 1 means activating the use of fixed acceleration, then the vehicle will ignore the desired acceleration computed by the CACC controller strategy. The third parameter is the acceleration we want to set. In leader crash attack, we set the acceleration -20 m/s2. According to Orders of magnitude (acceleration) [2], if an automobile crashes into an obstacle at the speed of 100 km/h, the acceleration can achieve -982 m/s2. In this lab, -20 m/s2 is enough to show leader crash attack. Besides, we need to choose the time when we want to launch the attack and which vehicle to launch the attack. GetInstance()→getCurrentTimeStep() returns the current time and veh→getID() return current vehicle’s id. Leader vehicle is platoon.0 .

 **Figure 4**

This picture shows that vehicle 1 and vehicle 2 all crash into vehicle 0 (leader car). Moreover, we can use the data collected by OMNeT++ to show the speed change for each vehicle. You can get the following picture by using R which has been introduced in *Plexe Version 1.1 Documentation.*

**Figure 5**

From Figure 5, we can see that line 1(blue) and line 2(green) terminate at 50.00 s when you launch leader crash attack. Line 4 and Line 5 represent new cars which show up after vehicle 1 and vehicle 2 crash into vehicle 1. It is a simulator bug and you can ignore this.

**3.4 Task 4: Collision Induction Attack**

In this attack[3], the attacker broadcasts an acceleration profile indicating that they are speeding up which causes the following vehicle to accelerate. The attacker actually starts to aggressively brake. This is very likely to cause an accident at high speed which makes this attack extremely dangerous.

To conduct this attack, we need to modify MSCFModel\_CC.cpp. The file path is /src/plexe-sumo/src/microsim/cfmodels/MSCFModel\_CC.cpp.

 **Figure 6**

 **Figure 7**

First, In Figure 6, we make vehicle 1 starts to decelerate at an acceleration of -9 m/s2 . We want it to decelerate to 80 km/h and maintain the speed. At the same time, vehicle 1 broadcasts false message to vehicle 2. The false message will finally go to MSCFModel\_CC.cpp, so we directly add codes in this file as shown in Figure 7. When controller computes the desired acceleration for vehicle 2, the speed of preceding vehicle we pass actually is 2\*predSpeed. Then the desired acceleration will be positive and vehicle 2 will follow false message to speed up.

 **Figure 8**

Figure 8 shows that vehicle 2 and vehicle 3 will follow the false message to speed up and then crash into vehicle 1. Vehicle 4 and 5 are system bugs and you can ignore them.

**3.5 Task 5: Message Falsification Attack**

In this attack, malicious driver does not send the real speed to following vehicle. Instead the attacker broadcasts false message to lead following vehicles to speed up or slow down. This attack is a little similar to collision induction attack except that malicious driver does not need to decelerate. For this attack, we also need to modify MSCFModel\_CC.cpp. The file path is /src/plexe-sumo/src/microsim/cfmodels/MSCFModel\_CC.cpp.

 **Figure 9**

Figure 9 is actually similar to Figure 7 except that this time we pass preSpeed/2 to vehicle 2. Vehicle 1 sends false message to tell following vehicles to slow down.

 **Figure 10**

Figure 10 shows that vehicle 2 follows false message to slow down and vehicle 3 crashes into vehicle 2. The reason why vehicle 2 speed up at the end is that when the gap is too big, the vehicle will be detached from the platoon and it will run by itself according to the controller. Line 4 is system bug and you can ignore it.

**4 Submission**

You need to submit a detailed lab report to describe what you have done and what you have observed; you also need to provide explanation to the observations that are interesting or surprising.

**5 References**

[1] Segata, Michele, et al. "Plexe: A platooning extension for Veins."*2014 IEEE Vehicular Networking Conference (VNC)*. IEEE, 2014.

[2] Orders of magnitude (acceleration) [https://en.wikipedia.org/wiki/Orders\_of\_magnitude\_(acceleration](https://en.wikipedia.org/wiki/Orders_of_magnitude_%28acceleration))

[3] DeBruhl, Bruce, et al. "Is your commute driving you crazy?: a study of misbehavior in vehicular platoons." *Proceedings of the 8th ACM Conference on Security & Privacy in Wireless and Mobile Networks*. ACM, 2015.