ATTACKING PLATOON - SIMULATION

Cyber Physical Systems and IoT Security - 2022/2023

In this laboratory, we use the Plexe simulator to visualize the effects of the jamming attack against platooning communications.

You need:
- VirtualBox
- The virtual machine

At https://drive.google.com/file/d/1k_4s2fzaQh2Ycr4d9cOjKF-CUVhn-urG/view?usp=share_link you can download the Ready-To-Use Plexe VM.

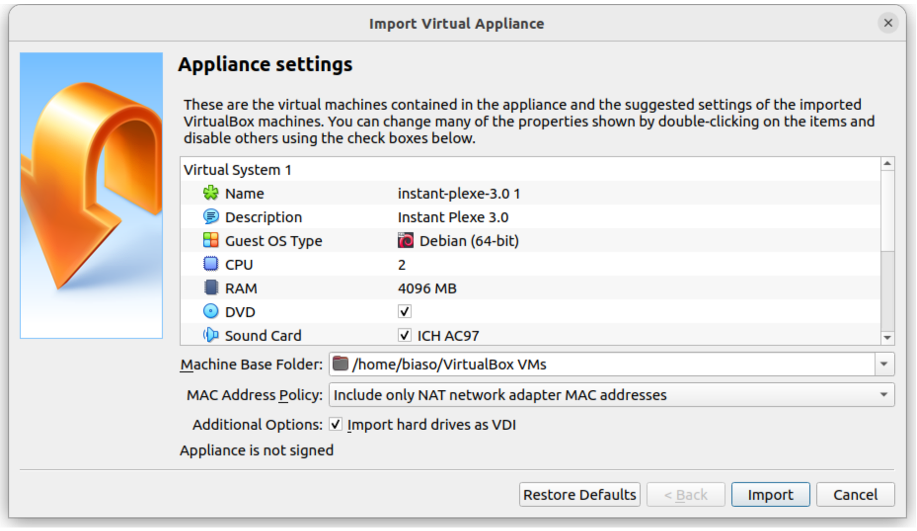
Plexe

Plexe is a platooning simulator based on the network simulator **Omnet++**, the traffic simulator **Sumo** and an additional comprehensive vehicular simulator **Veins** (Plexe is actually an extension of this last one).

You can find all the information and tutorial at: https://plexe.car2x.org/

Installation and useful files

Once you downloaded the VM, you just need to double-click on the *.ova* file to import it in Virtual Box. Then confirm with the *import* button.



Appliance settings

These are the virtual machines contained in the appliance and the suggested settings of the imported VirtualBox machines. You can change many of the properties shown by double-clicking on the items and disable others using the check boxes below.

Virtual System 1

Name	instant-plexe-3.0 1
Description	Instant Plexe 3.0
Guest OS Type	Debian (64-bit)
CPU	2
RAM	4096 MB
DVD	☑
Sound Card	☑ ICH AC97

Machine Base Folder: /home/biaso/VirtualBox VMs

MAC Address Policy: Include only NAT network adapter MAC addresses

Additional Options: ☑ Import hard drives as VDI

Appliance is not signed

Import window.

Now that you have the VM up and running, you can just open a Files window. Notice: in the VM you don't have any icon on the Desktop, click on the *Activities* voice in the upper-left corner of the screen. Another option is to open a terminal e navigate through the folders with the usual commands ($ls*, *cd$).

Now you can go to the *src* folder that contains all the tools and source code that we need.

```
$cd src
```

Here, the importat files are in the examples in *plexe* folder:

```
$cd plexe/examples
```

In this folder, you have different sub-folders containing various simulation files. You can also try to run some of them if you want.
Today, we want to see the attack simulations that are in *my_attacks* folder.

N.B. The *attacks* folder is the one from which I took the base code. You don't need to look at it. It is not properly working due to old dependencies).

Now, the simulation file is the one called **omnetpp.ini**, where all the parameters and general configuration are defined.

This is a snippet of the code inside this file, managing the traffic parameters.

```
##########################################################
# Traffic manager #
##########################################################

**.traffic_type = "PlatoonsTrafficManager"
#insert platooning vehicles at time
**.traffic.platoonInsertTime = 1 s
#insert platooning vehicles with a speed of
**.traffic.platoonInsertSpeed = ${leaderSpeed}kmph
#insert nCars platooning vehicles
**.traffic.nCars = ${nCars}
#let platoonSize cars per platoon
**.traffic.platoonSize = ${platoonSize}
#use nLanes lanes
**.traffic.nLanes = ${nLanes}
#SUMO vtype for platooning vehicles
**.traffic.platooningVType = "vtypeauto"
#insert vehicles already at steady-state.
#distance depends on controller


#disable statistics recording for all other modules
**.scalar-recording = false
**.vector-recording = false
```

Run the simulator

To start a simulation, first run this command inside the **plexe** folder:

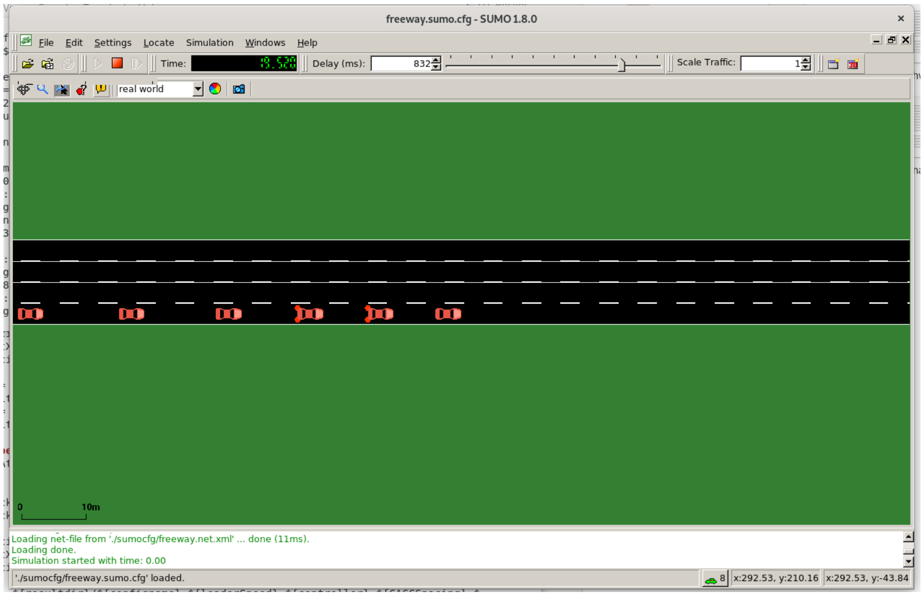
```
$cd ~/src/plexe
$. ./setenv
```

Return to the **my__attacks** folder:

```
$cd examples/my_attacks
```

Now you are ready to simulate an environment with the command **plexe__run**:

```
$plexe_run -u Cmdenv -c Sinusoidal
```



```
Loading net-file from './sumocfg/freeway.net.xml' ... done (11ms).
Loading done.
Simulation started with time: 0.00
'./sumocfg/freeway.sumo.cfg' loaded.
```

Plexe simualtion with no attack ongoing.

This simulation doesn't present any attack, but you can use it to familiarize with the simulator. It starts an instance of Sumo where you can see the traffic and a sinusoidal behaviour between vehicles of the same platoon.

The othe configurations in the *.ini* file are delineated by the following nomenclature

```
[Config nameOfTheScenario]
```

The simulation

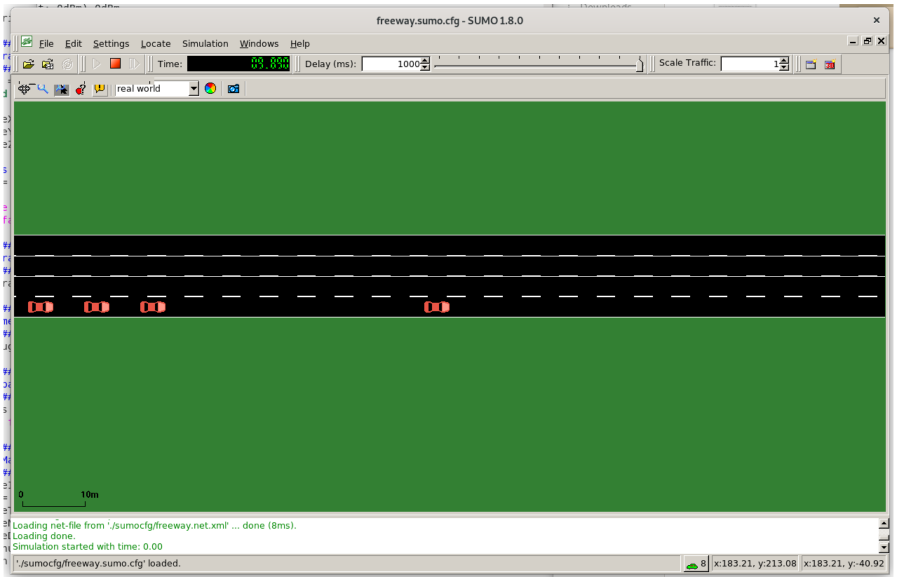
Perfect, now you can run the attack simulation starting with

```
[Config SinusoidalJammingAttack]
```

with the command

```
$plexe_run -u Cmdenv -c SinusoidalJammingAttack
```

and see what happens.



Time: 09.890

Delay (ms): 1000

Scale Traffic: 1

```
Loading net-file from './sumocfg/freeway.net.xml' ... done (8ms).
Loading done.
Simulation started with time: 0.00
```

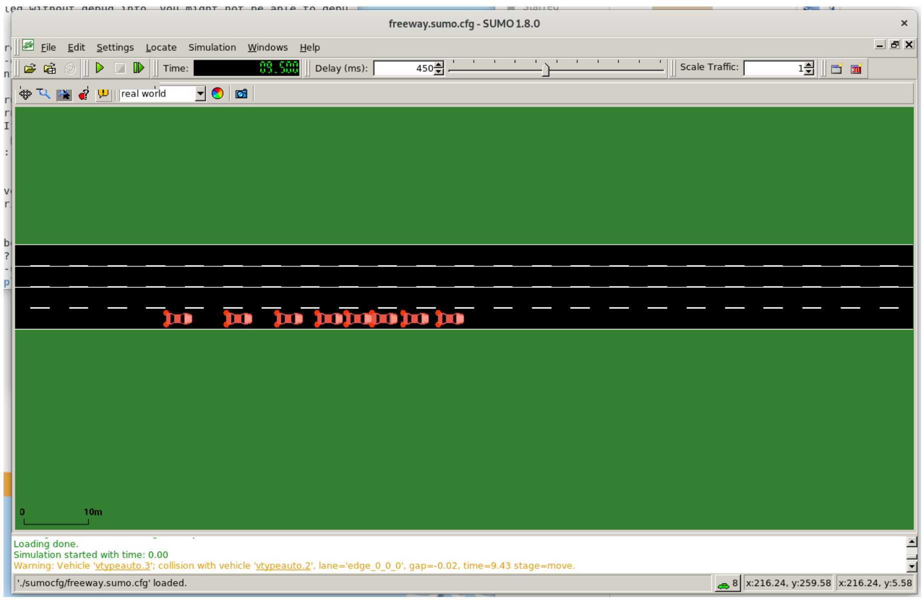
'./sumocfg/freeway.sumo.cfg' loaded. 8 x:183.21, y:213.08 x:183.21, y:-40.92

Jamming attack against the communication in the platoon, the head of the platoon just "goes South".

Another interesting scenario is when the communication during a braking phase is not successful:

```
[Config Braking]
```

```
$plexe_run -u Cmdenv -c Braking
```



Time: 09.500
Delay (ms): 450
Scale Traffic: 1

```
Loading done.
Simulation started with time: 0.00
Warning: Vehicle 'vtypeauto.3'; collision with vehicle 'vtypeauto.2', lane='edge_0_0_0', gap=-0.02, time=9.43 stage=move.
'./sumocfg/freeway.sumo.cfg' loaded.
```

8 x:216.24, y:259.58 x:216.24, y:5.58

Accident during braking phase in the platoon.

Results and Plots

The VNC paper provides some scripts for the analysis, in particular we need to use the *injectionCrashStats.py* script, for example:

```
python3 injectionCrashStats.py
--infolder raw_data_new/SinusoidalPosInjection/
--plotfolder graphs_new --plotname 'SinusoidalPosInjection'
--csvout SinusoidalPosInjection-new.csv --plotDeltaV
```

This parsers doesn't work completely with the new format of the data, so this demostration uses the files provided by the authors. I'm working on the analysis of the results in the new updated Plexe software.

Create your own

From the Plexe tutorial page (on their website) and the files in our presentation, you can try to roll your own simulation. A great part of the code can be the same and you can just try to play with some parameters (e.g. number of cars, lanes, manouvers).

References

1. https://plexe.car2x.org/

2. https://github.com/vs-uulm/plexe-veins/tree/vnc2017-cacc-attack-analysis
3. https://github.com/vs-uulm/vnc2017-CACC-data

Contacts

If you need anything about this lab, you can send an email to:
tommaso.bianchi@phd.unipd.it