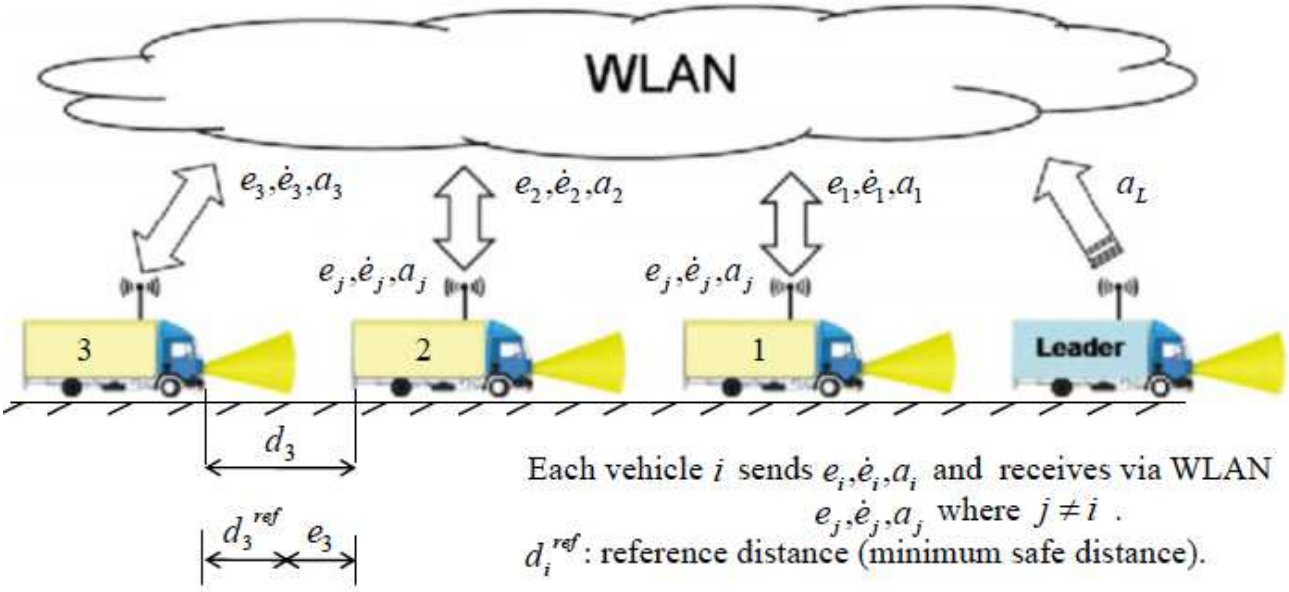
set trace onto (x_1, x_2) subspace. The system starts evolving in time in free-flow mode from a set of initial conditions at $t = 0$, whose boundary is shown in magenta. The free-flow reach set evolving from $t = 0$ to $t = 100$ is shown in blue. Between $t = 18$ and $t = 68$ the free-flow reach set crosses the guard. The guard is shown in red. For each nonempty intersection of the free-flow reach set and the guard, the congested mode reach set starts evolving in time until $t = 100$. All the congested mode reach sets are shown in green. Observe that in the congested mode, the density x_2 in the congested part decreases slightly, while the density x_1 upstream of the congested part increases. The blue set above the guard is not actually reached, because the state evolves according to the green region.

LMI-based three-vehicle platoon

A platoon of vehicles includes typically a leader and a number of followers. In a controlled platoon the controllers are designed to maintain constant relative distances between autonomous vehicles by tracking the trajectory of the leader. The latter is manually driven and can be considered as a reference input to the whole system. We are concerned with the longitudinal control of a platoon of vehicles engaged in following each other longitudinally by exchanging information via a wireless local area network (WLAN) (see Fig.1).

. _platoonfig1:



alt: platoon
width: 50 %
Platoon struture and notations.

The spacing errors :math: *e_i* are defined as the difference between the actual distance to the predecessor and a (fixed) reference distance: :math: *e_i(t) = d_i(t) - dref_i.* Bounds for these reference distances will be stipulated by the result of our safety verification. The effective acceleration ai of each vehicle within the platoon is governed by the drivetrain dynamics.According to Fig. 1 and with the further approximation the resulting platoon model is given by:

$$t = 100$$

label: platoon1

ddot{e}_i &= a_{i−1} - a_i.

$$t = 100$$

label: platoon2

a_i &= -1frac{1}{tau_i}a_i + 1frac{1}{tau_i}v_i

where τi is the time constant of the drivetrain considered here to be constant and vi the input signal. The dynamics of the whole platoon with a state vector :math: *x =[… e_i, dot{e}_i, a_i …]^T* can be expressed in state space form as follows:

$$t = 100$$

label: platoon3

dot{x} &= A_sx+B_1a_L+B_2v.

where the leading vehicle's acceleration aL enters the dynamics as a disturbance. The goal thereby is to stabilize the platoon and realize a good disturbance rejection in terms of small spacing errors at reasonable control effort. These constraints comprise in particular maximum (absolute) spacing errors to prevent collisions among platoon members but also maximum amplification of velocity or acceleration values to account for the existing saturation effects that arise due to force limitation between road and tire. This optimal control problem is applied to a state feedback structure assuming that each vehicle has information access to all other vehicles states. We obtain as result an optimal matrix K verifying:

$$t = 100$$

label: platoon4

v &= K x

The closed loop system is hence given by:

$$t = 100$$

label: platoon5

dot{x} &= Ax + Bu. where :math: *A = (A_s + B_2K), B = B_1* and :math: *u = a_L*.

The main goal of this work, is to investigate the impact of disturbances of the communication network on the performance of the cooperative platoon. We are particularly interested in worst cases, in which a loss of communication between two/many or all vehicles occurs. The theory of hybrid systems offers a convenient framework to model this kind of systems. A hybrid automaton consists of states described by continuous dynamics and discrete events which trigger transitions between these states. Our application can be modeled by a hybrid automaton. The controlled platoon dynamics constitute thereby the continuous states and the communication breakdowns trigger the discrete switches from one continuous state to another. The interconnection topology within the platoon is modeled with a directed graph G = (V,E), defined by vertices V and edges E. The ith vertex represents the ith vehicle and the edge (i, j) indicates that vehicle j receives information from vehicle i. This graph is represented by the adjacent matrix :math: *R = [r_{ij}]* referred to as the communication matrix of the platoon.

To take into account the communication failures in the controller design, the loss of information is expressed by forcing zeros in K. Depending on the topology and the configuration of the communication between vehicles given by the matrix R, many communication scenarios are possible. Consequently, the hybrid automaton modeling this kind of system will be complex. We focus our study on safetycritical worst case scenarios.We consider the worst case in Fig.2, in which the system switches from a full to a total dropout of the communication between the vehicles within the platoon. In general, our controlled hybrid automaton has continuous states.

. _platoonfig2:

alt: automat
width: 50 %

Hybrid automata modeling the worst case scenario.

To each continuous state q corresponds a new K_q and consequently new matrices A_q and B_q verifying the equation .. math:: :label: platoon6 dot{x}(t) = A_qx(t) + B_qu(t)

where :math: *x(t) in R^9* denotes the state vector, :math: *u(t) = a_L in R* is

the input vector and :math: *q in {1, 2}* is the mode described by :math: *(A_q,B_q) in R^{9 multiply 9} multiply R^9.*

```
%%
%LMI-based three-vehicle platoon

%system parameters
%elltool.setconf('default');
elltool.conf.Properties.setNTimeGridPoints(150);
%elltool.conf.Properties.setRelTol(1e-3);

%initial conditions(1)

% define system 1

%In case of no communication problems, these matrices are given as follows
firstAMat = [     0      1      0       0      0       0       0      0       0
                  0      0     -1       0      0       0       0      0       0
             1.6050 4.8680 -3.5754 -0.8198 0.4270 -0.0450 -0.1942 0.3626 -0.0946
                  0      0      0       0      1       0       0      0       0
                  0      0      1       0      0      -1       0      0       0
             0.8718 3.8140 -0.0754  1.1936 3.6258 -3.2396 -0.5950 0.1294 -0.0796
                  0      0      0       0      0       0       0      1       0
                  0      0      0       0      0       1       0      0      -1
             0.7132 3.5730 -0.0964  0.8472 3.2568 -0.0876  1.2726 3.0720 -3.1356];


firstBMat = [0 1 0 0 0 0 0 0 0]';

%if we want to specify the interval [a, b]
%for control constraints a_L
```

```
a = 2;
b = 9;

firstUBoundsEllObj = ellipsoid((b+a)/2,(b-a)/2);

% define system 2

%In case of total disruption of the communication, the matrices describing
%the system are given by
secAMat = [    0 1.0000       0       0      0       0       0      0       0
               0      0 -1.0000       0      0       0       0      0       0
          1.6050 4.8680 -3.5754       0      0       0       0      0       0
               0      0       0       0 1.0000       0       0      0       0
               0      0  1.0000       0      0 -1.0000       0      0       0
               0      0       0  1.1936 3.6258 -3.2396       0      0       0
               0      0       0       0      0       0       0 1.0000       0
               0      0       0       0      0  1.0000       0      0 -1.0000
          0.7132 3.5730 -0.0964  0.8472 3.2568 -0.0876  1.2726 3.0720 -3.1356];


secBMat = [0 1 0 0 0 0 0 0 0]';

%if we want to specify the interval [a, b]
%for control constraints a_L

secUBoundsEllObj = ellipsoid((b+a)/2,(b-a)/2);

%time options

% time horizon T after which the controlled system reaches a stable state

T = 1.2;


%switching time from the first discrete state

switchTimeFirst = 0.1;
switchTimeSec = 1;

%e_i options
e_1 = 0;
e_2 = 0;
e_3 = 0;

%vector that shows that we start without collisions
% set of initial conditions

 VecIn = [2 0 0 2 0 0 2 0 0];
```

language: matlab
linenos:

```
%%
% programm

firstSys = elltool.linsys.LinSysContinuous(firstAMat, firstBMat,firstUBoundsEllObj);

x0EllObj = ellipsoid(VecIn',eye(9));

% columns of L specify the directions
dirsMat = [1 0 0 0 0 0 0 0 0
           0 1 0 0 0 0 0 0 0
           0 0 1 0 0 0 0 0 0
           0 0 0 1 0 0 0 0 0
           0 0 0 0 1 0 0 0 0
           0 0 0 0 0 1 0 0 0
           0 0 0 0 0 0 1 0 0
```

```
          0 0 0 0 0 0 0 1 0
          0 0 0 0 0 0 0 0 1]';
                    firstRsObj = elltool.reach.ReachContinuous(firstSys, x0EllObj, dirsMat,..
                                                               [0 switchTimeFirst], 'isRegEna
                                                               'regTol', 1e-5,'absTol',1e-6,'

% solve collision with same times
secSys = elltool.linsys.LinSysContinuous(secAMat, secBMat,secUBoundsEllObj);
 if switchTimeSec == switchTimeFirst
     thRsObj = firstRsObj.evolve(T, firstSys);
 else
     secRsObj = firstRsObj.evolve(switchTimeSec, secSys);
     thRsObj = secRsObj.evolve(T, firstSys);
 end


basisMat1 = [1 0 0 0 0 0 0 0 0
             0 0 0 1 0 0 0 0 0]';

basisMat2 = [1 0 0 0 0 0 0 0 0
             0 0 0 0 0 0 1 0 0]';

basisMat3 = [0 0 0 1 0 0 0 0 0
             0 0 0 0 0 0 1 0 0]';

thPsObj1 = thRsObj.projection(basisMat1);
thPsObj2 = thRsObj.projection(basisMat2);
thPsObj3 = thRsObj.projection(basisMat3);

% external approximation
plObj=smartdb.disp.RelationDataPlotter('figureGroupKeySuffFunc', ...
    @(x)sprintf('_forward_reach_set_proj%d',x));
thPsObj1.plotByEa('r',plObj);

%%
```

language: matlab
linenos:

. _platoonfig3:

alt: automat
width: 50 %

. _platoonfig4:

alt: automat
width: 50 %
<h2>References</h2>

[SUN2003] *(1, 2)* L.Muñoz, X.Sun, R.Horowitz, and L.Alvarez. 2003. Traffic Density Estimation with the Cell Transmission Model. In *Proceedings of the American Control Conference*, 3750–3755. Denver, Colorado, USA.