

Towards Adaptive Fault Tolerance on ROS for Advanced Driver Assistance Systems



Matthieu Amy

Jean-Charles Fabre, Michael Lauer

Toulouse, France

Context and trends

From ADAS to autonomous driving, e.g. ACC (Adaptive Cruise Control), TJP (Traffic Jam Pilot)..... Etc.

Agile Development Process....
Rapid prototyping... meaning
Short validation time...

Remote dynamic updates,
maintenance, improvements,
news features... novel business



Tesla vehicles regularly receives **over-the-air software updates** that add new features and functionality. When an update is available, you'll be notified on the center display with an option to install immediately, or schedule the installation for a later time. Connect your vehicle to your home's Wi-Fi network for the fastest possible download time.

Context and trends

From ADAS to autonomous driving, e.g. ACC (Adaptive Cruise Control), TJP (Traffic Jam Pilot)..... Etc.



**Safety critical system...
stringent dependability issues
despite fast evolution!**

Tesla vehicles regularly receives *over-the-air software updates* that add new features and functionality. When an update is available, you'll be notified on the center

Resilient Computing: persistence of dependability despite changes

Motivations and objectives of our **on-going work!**

Fast evolution, Agile Dev. , time to market,
Over-the-Air updates.....

Problem statement and key concepts

Once the system is deployed, it faces changes due to maintenance or evolution.

System designers cannot predict everything in advance....

Persistence of dependability requires the adaptation of safety mechanism

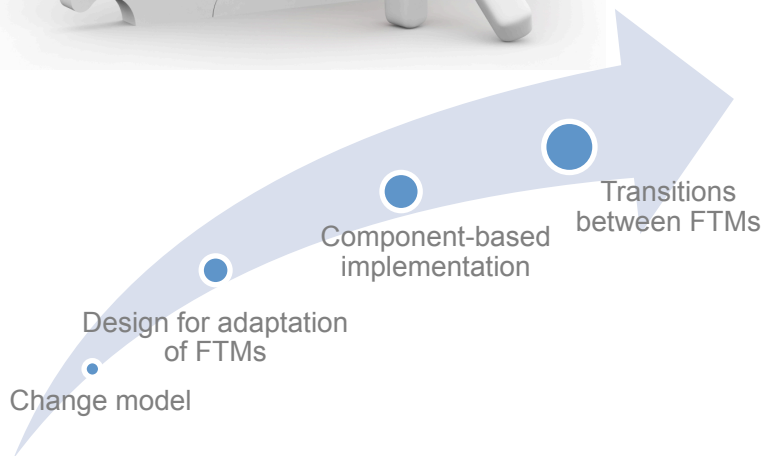
Key concepts for **Adaptive Fault Tolerance (AFT)**

- Separation of concerns
- Design for adaptation
- Remote fine-grained updates

Outline



- ◆ **Introduction to Adaptive Fault Tolerant Computing**
- ◆ **What runtime support for AFT as a *Lego* system: ROS?**
- ◆ **How to combine AFT with over-the-air updates of critical ADAS?**
- ◆ **A simple experimental platform**

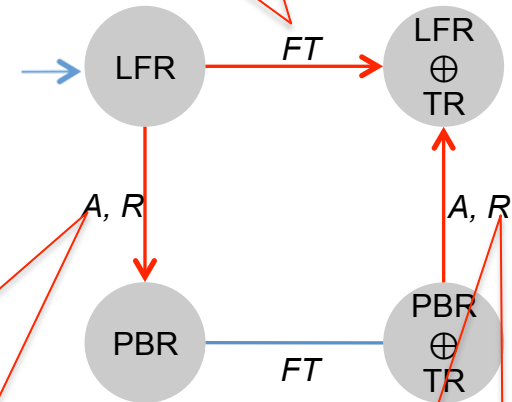


Assumptions and FTM Characteristics

TRANSITIONS

Assumptions / FTM		PBR	LFR	TR
Fault Model (FT)	Crash	✓	✓	
	Transient			✓
Application behaviour (A)	Deterministic		✓	✓
	State access	✓		
Resources (R)	Bandwidth	high	low	nil
	# CPU	2	2	1

Trigger: high rate of HW transient faults observed

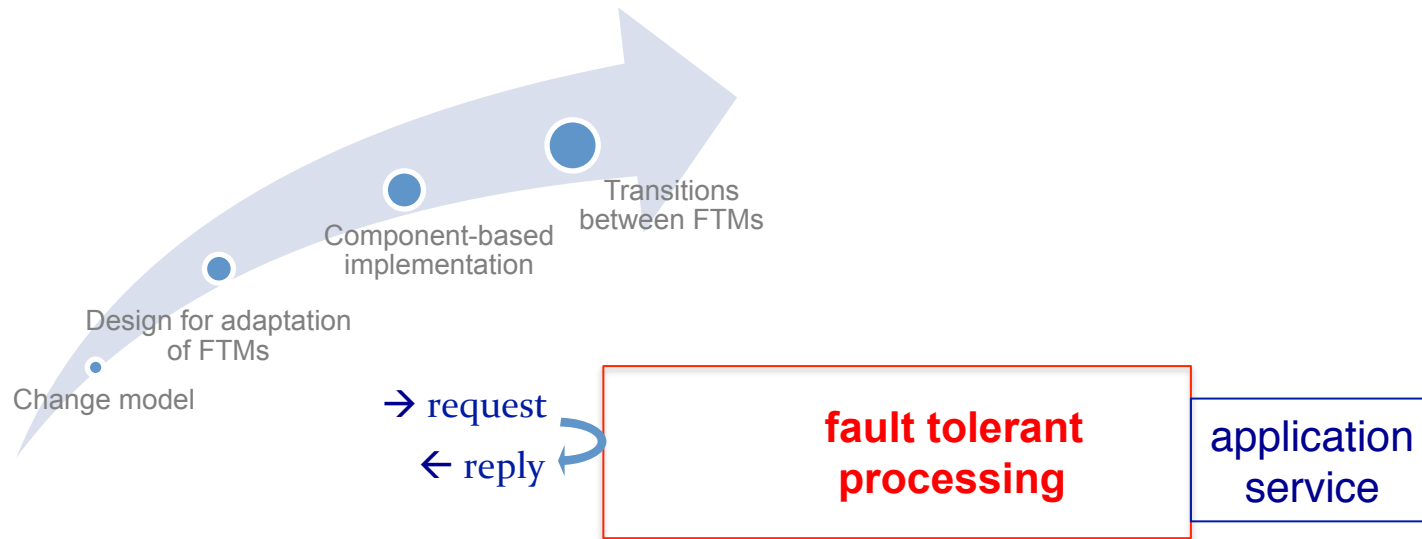


Trigger: Non deterministic SW application version

Trigger: bandwidth drop below a given threshold

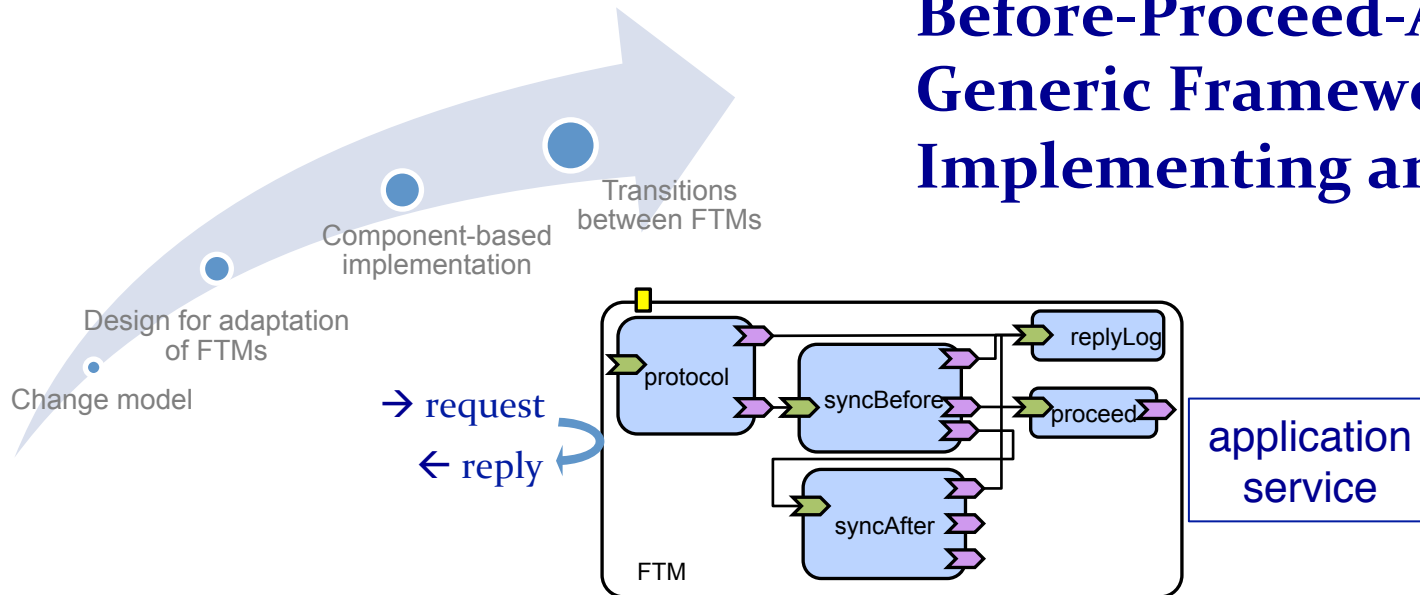
PBR=Primary-Backup Replication
 LFR=Leader-Follower Replication
 TR=Time Redundancy

Componentization of FTM



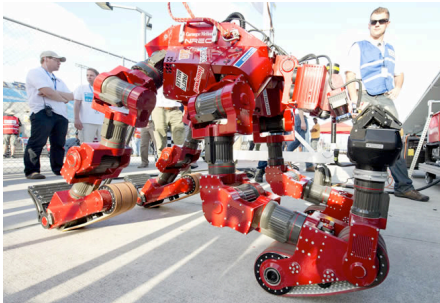
Componentization of FTM

Before-Proceed-After Generic Framework for Implementing any FTM



FTM	Before	Proceed	After
PBR (primary)		Compute	Checkpointing
PBR(backup)			State update
LFR (leader)	Forward request	Compute	Notify
LFR (follower)	Handle request	Compute	Handle notification
TR	Save/restore state	Compute	Compare

Is ROS a good candidate for AFT in automotive embedded systems?



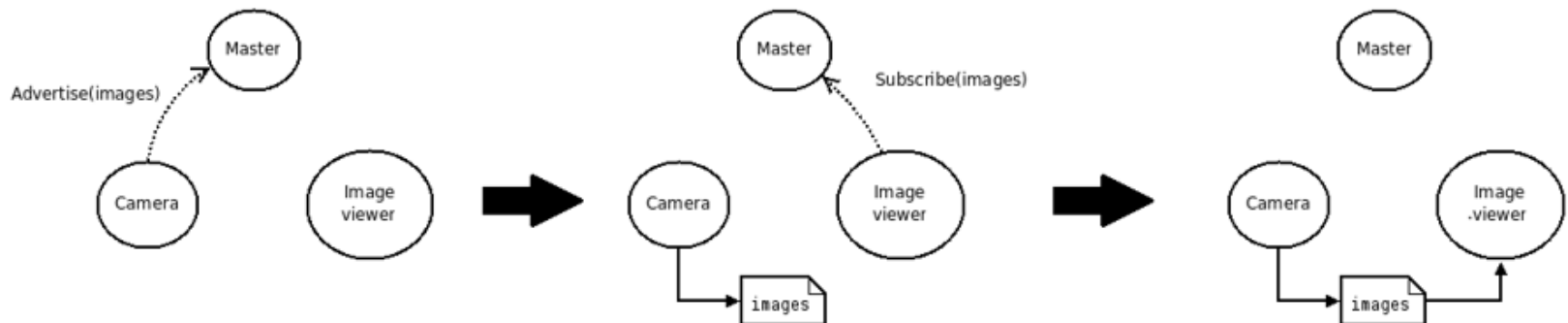
“**BMW** has been working on automated driving for the last decade, steadily implementing more advanced features ranging from emergency stop assistance and autonomous highway driving to fully automated valet parking and 360° collision avoidance. **Several of these projects were presented at the 2015 Consumer Electronics Show, and as it turns out, the cars were running ROS for both environment detection and planning.**” (Michael Aeberhard (BMW): *Automated Driving with ROS at BMW, May 31, 2016*)

What is ROS ?

Publish-subscribe middleware

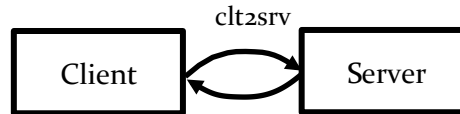
- Rosmaster : Communication master
- *Nodes* : isolated processes
- TCP/IP communication
 - ✓ Topic for asynchronous communications
 - ✓ Service for synchronous interaction

Implementation of a asynchronous communication (Topic)



Design for FTM adaptation on ROS

Generic computation graph for FTM (Boxes represent nodes)



- **Topics(0)**

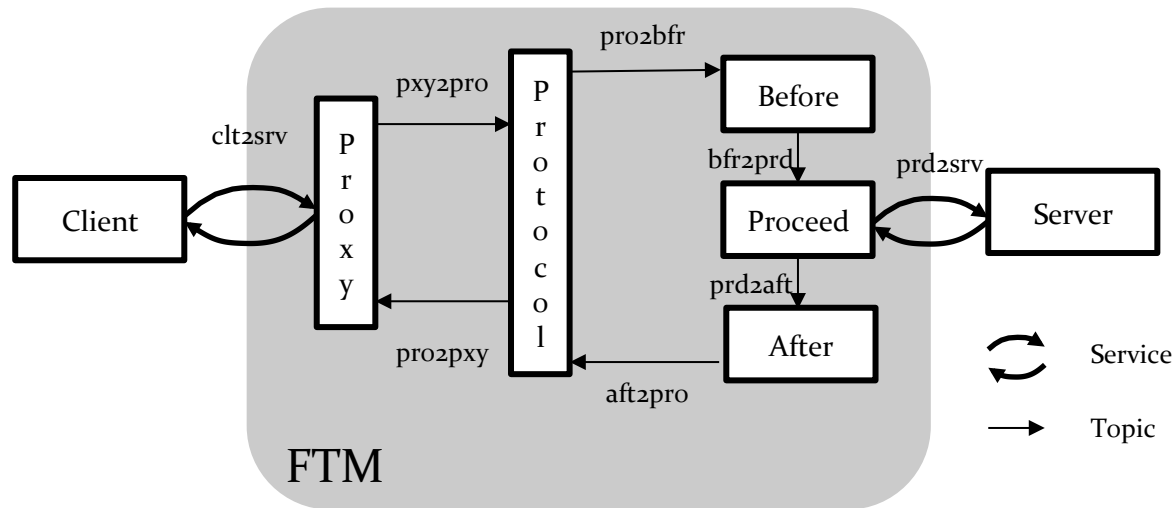
- **Nodes(2)**

- Client
- Server

Services: clt2srv (client to server)

Design for FTM adaptation on ROS

Generic computation graph for FTM (Boxes represent nodes)



- **Topics(6)**

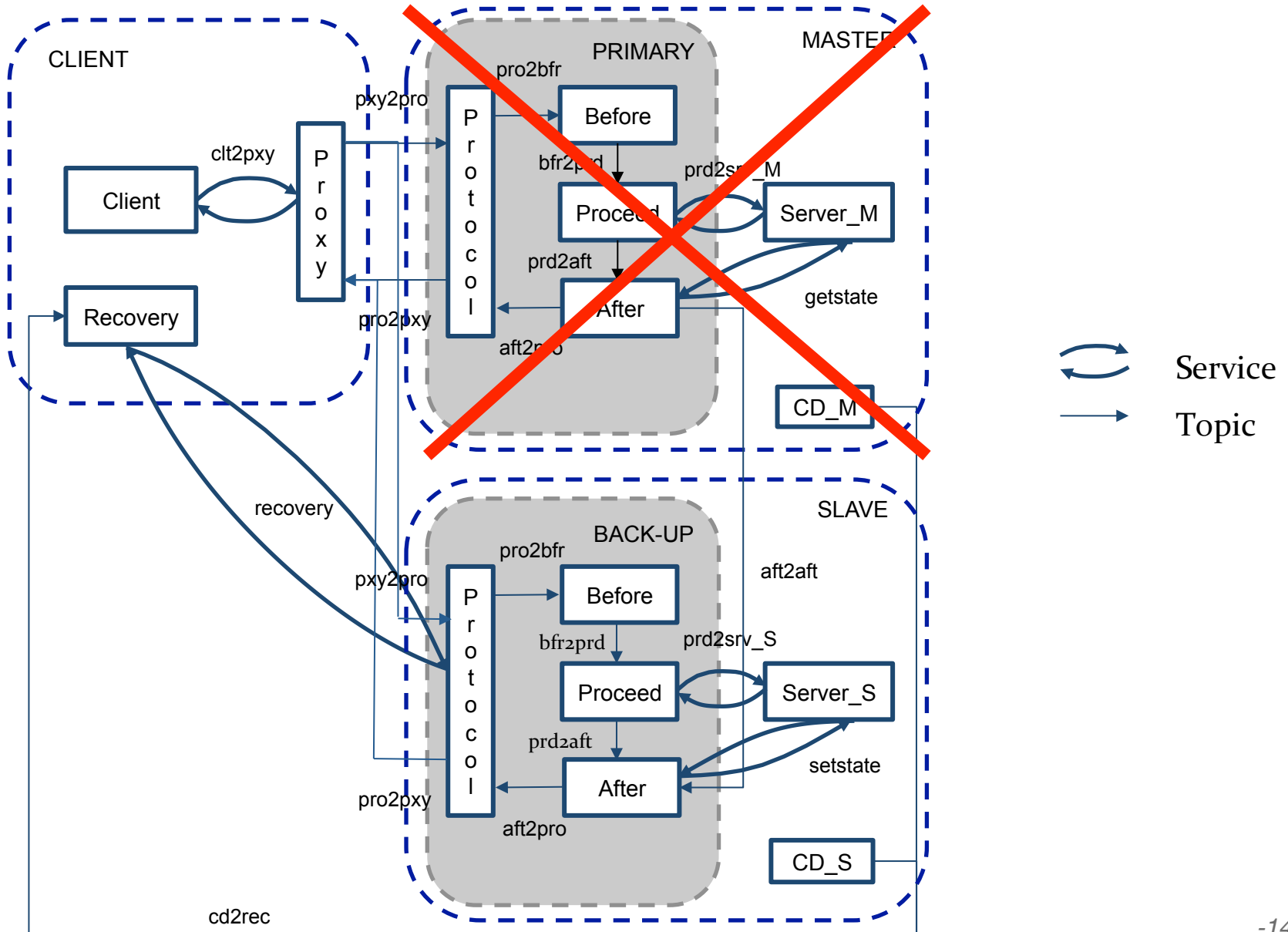
- pxy2pro
- pxy2bfr, bfr2prd, prd2aft
- aft2pro
- pro2pxy

- **Nodes(5+2)**

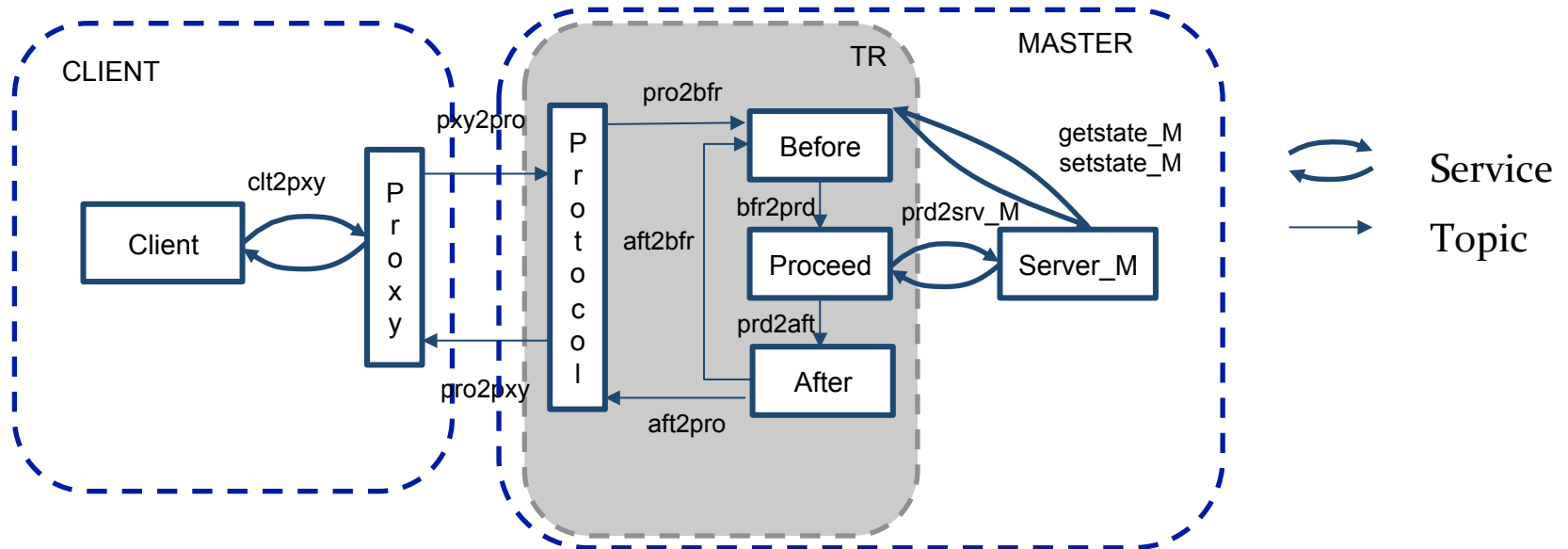
- Client
- Server
- Proxy
- Protocol
- Before, Proceed, After

Services: clt2pxy (client to proxy) and prd2srv (proceed to server)

Implementing PBR on ROS

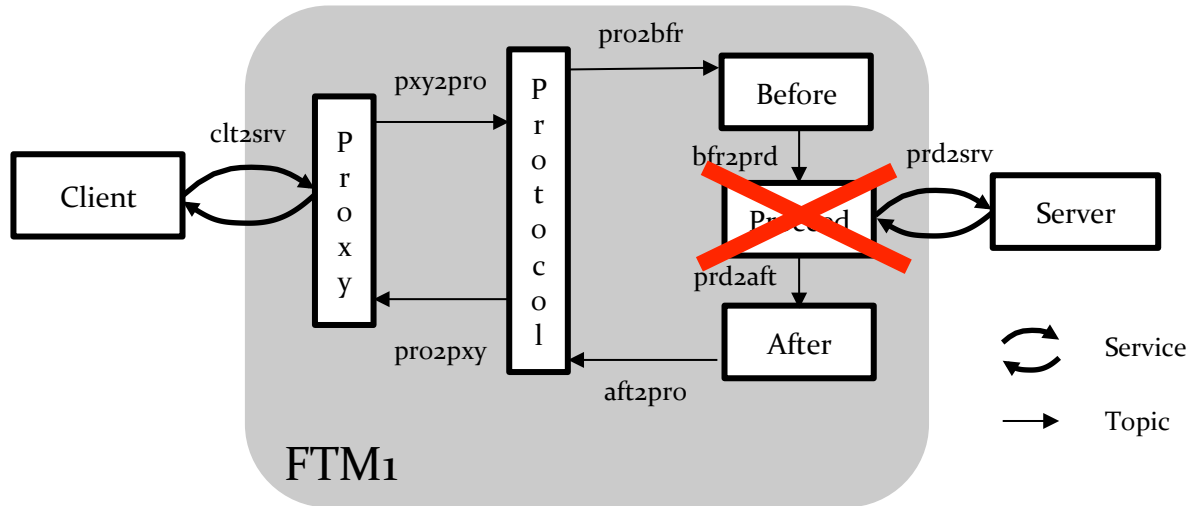


Implementing TR on ROS



Combining FTM on ROS

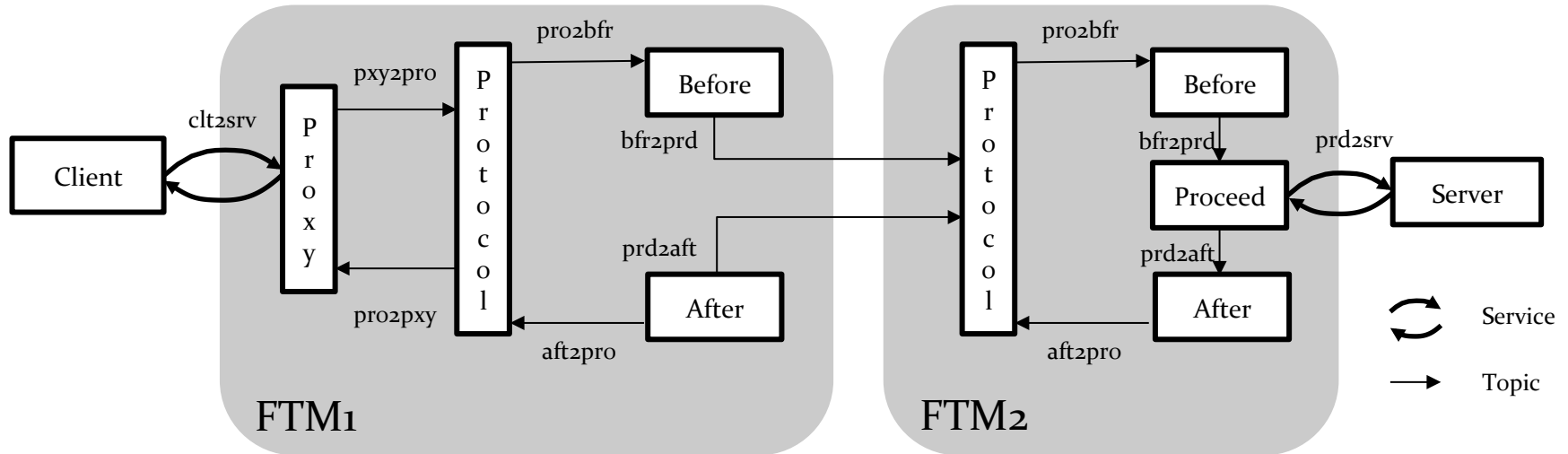
Generic composition graph for FTM



- **Protocol node is a software rack of nodes**
 - Before
 - Proceed → activation of services or protocols
 - After
- **Protocol node can substitute for proceed node**
 - It can be view as a frontend of the server...

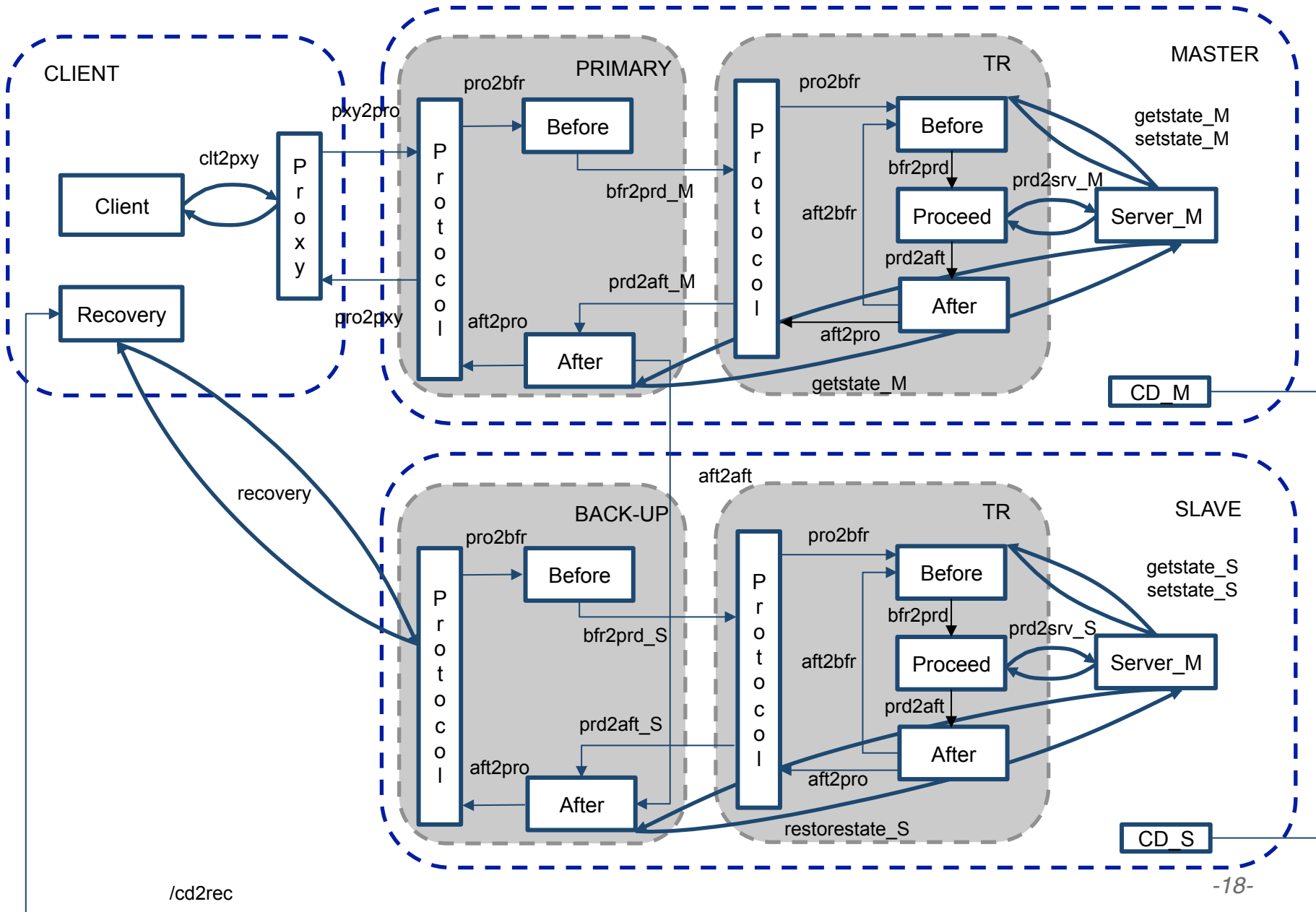
Combining FTM on ROS

Generic composition graph for FTM



- **Protocol node is a software rack of nodes**
 - Before
 - Proceed → activation of services or protocols
 - After
- **Protocol node can substitute for proceed node**
 - It can be view as a frontend of the server...

Combining PBR+TR on ROS



Lessons learnt

- **ROS nodes**
 - confinement area / space partitioning
 - Graph of nodes / active components
- **Node control:**
 - Manipulation of the nodes (add, remove),
 - Suspend/activate nodes done using *Unix Signals sent by an Adaptation Node*
 - Buffering of messages
- **Bindings**
 - Bindings at initialization only (notion of remapping).
 - Port management function added to nodes and invoked by a *Recovery Node as a service*

• Summary

Dynamicity of control and bindings solved using **ROS features + Unix Signal + additional logic** into the application nodes + **sysadmin nodes Adaptation and Recovery Nodes.**

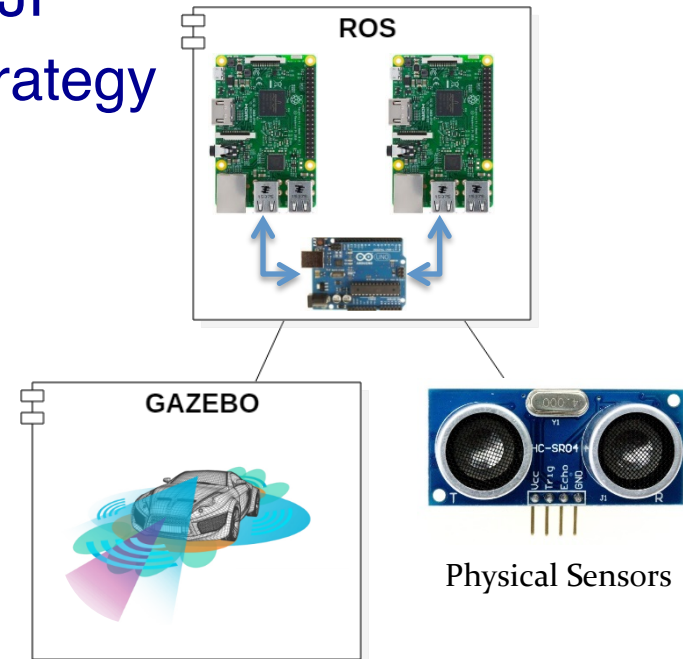
Experimental platform for development and validation of resilient ADAS

Why an experimental Platform

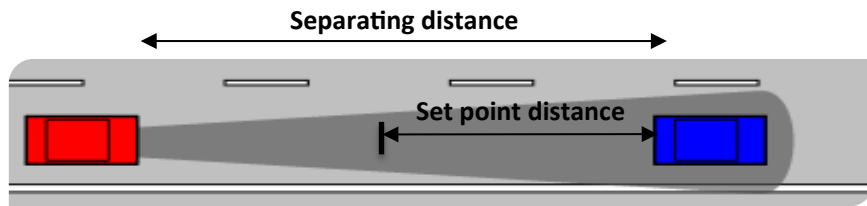
- **Motivation**
 - Development of a simulation platform for ADAS
 - Failure mode analysis using fault injection techniques
- **Status**
 - Development of Traffic Jam Pilot ADAS
 - Dependable computing architecture
- **On-going work**
 - *Over-The-Air* updates
 - Improvement / variants of the TJP
 - Dynamic reconfiguration of FTM
 - Validation of by fault injection

Global Platform Architecture

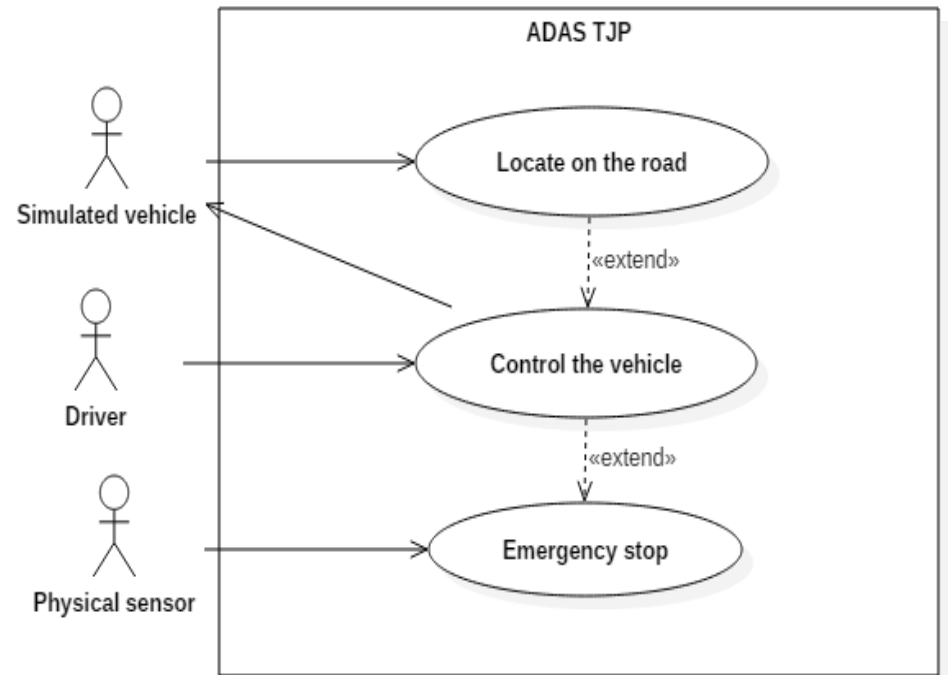
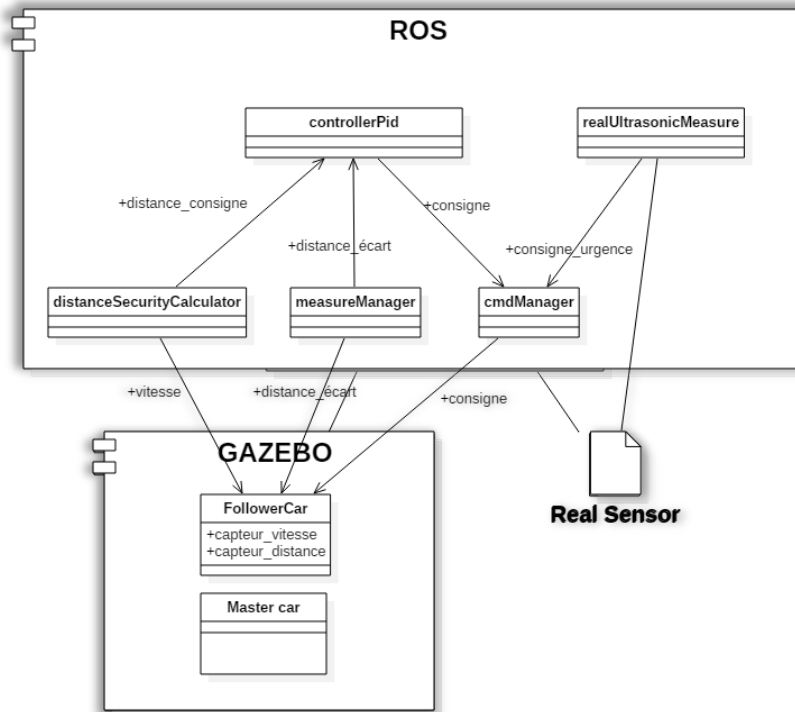
- **Platform**
 - ROS Implementation of the TJP
 - Duplex architecture and FT strategy
- **Gazebo: 3D simulator**
 - The Car dynamics
 - Virtual sensors
- **Real sensors**



Use Cases – TJP ADAS



The TJP automatically adjusts the speed of the follower car to maintain a safe distance to the master one

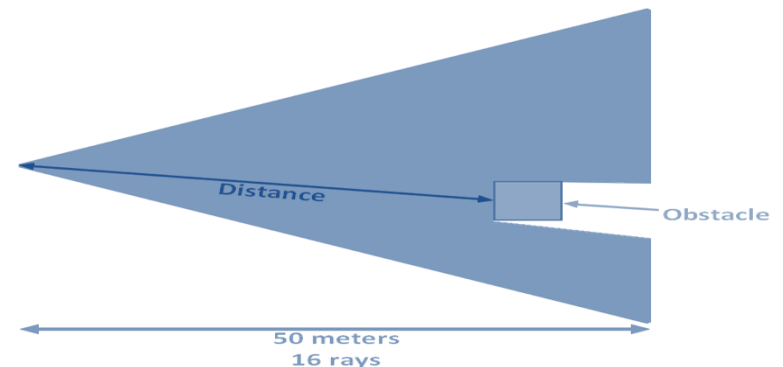
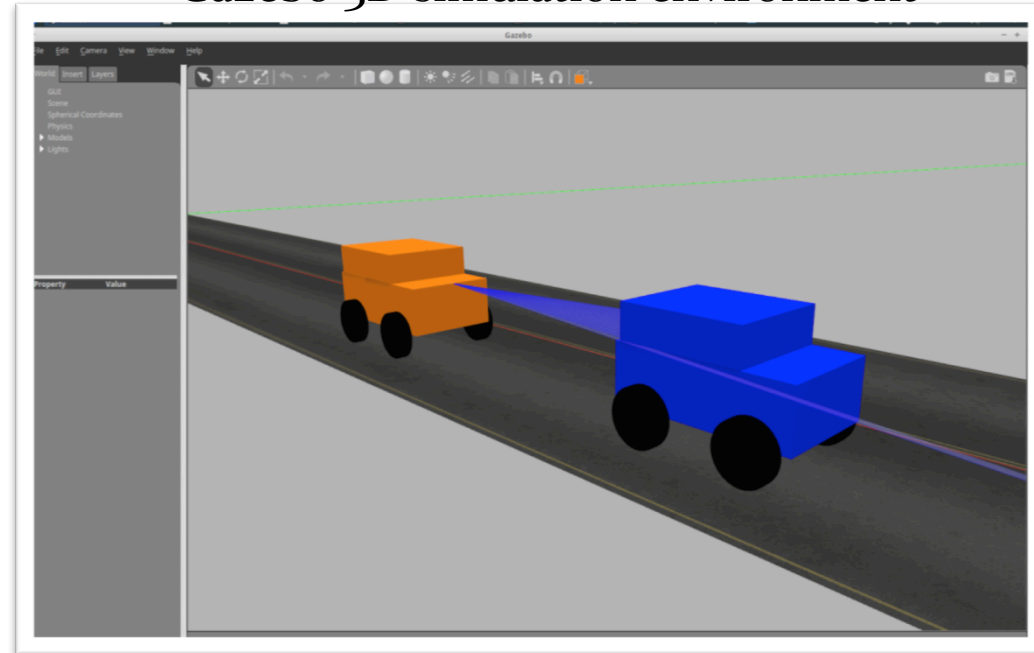


Simulation with Gazebo



Gazebo 3D simulation environment

- **TJP with two cars**
 - Master simulating a traffic jam
 - **Follower** with sensors controlling a distance
- **Plugins: Sensors (*Follower only*)**
 - Laser sensor (*distance*)
 - Inertial Measurement Unit (*speed*)

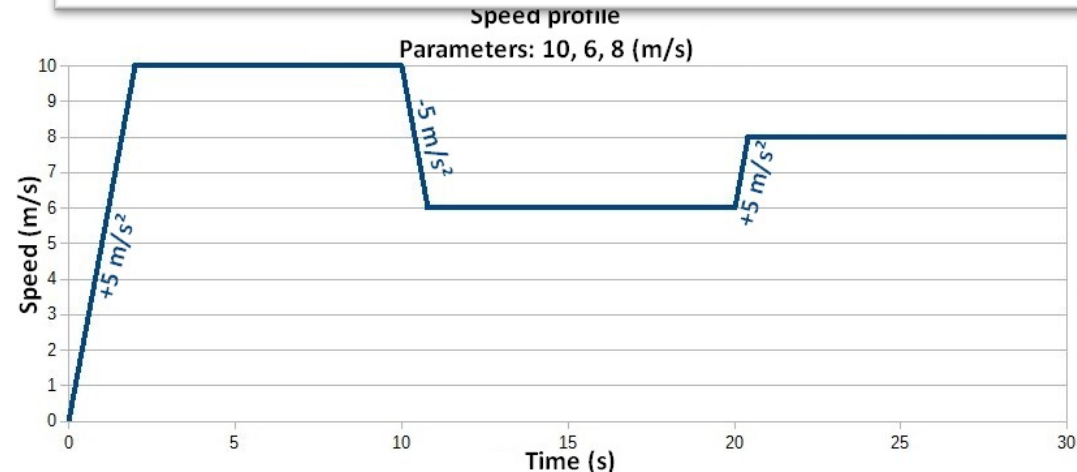
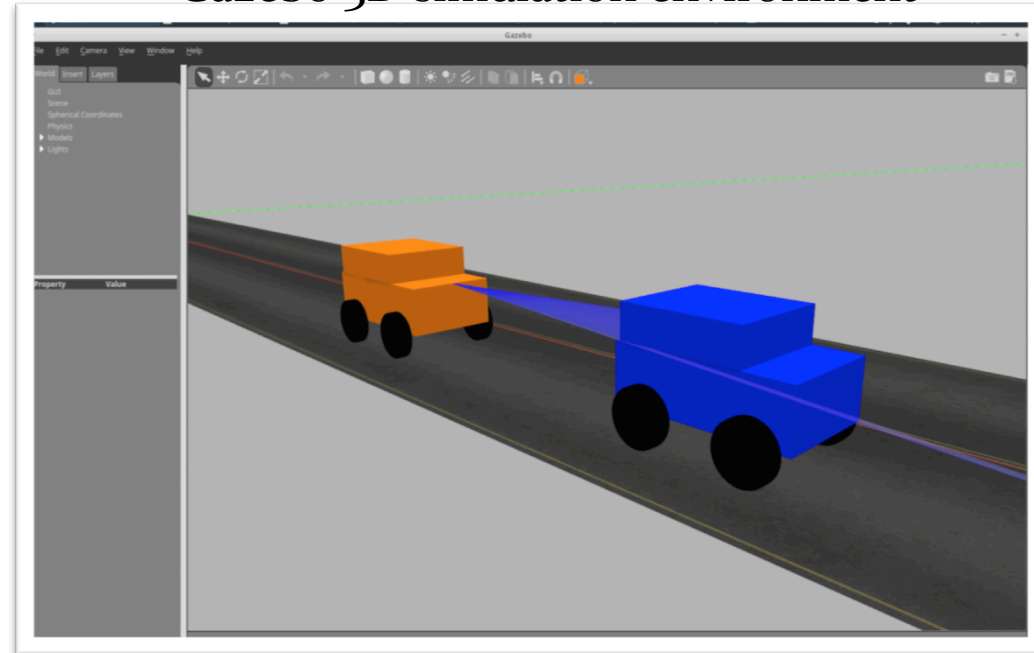


Simulation with Gazebo



Gazebo 3D simulation environment

- **TJP with two cars**
 - Master simulating a traffic jam
 - **Follower** with sensors controlling a distance
- **Master**
 - Speed profile
- **Follower**
 - Speed set point



Functional validation

	Entity	Input	Output	Test
ROS	distanceSecurityCalculator	Current Speed	Set point distance	Give a speed value, check the set point distance calculated
	controllerPID	Set point distance + Separating distance	Speed command	Check the speed value
	cmdManager	Speed command	Speed command	Check the priority management
	realUltrasonicSensor	Real distance	Speed Command	Check the data read by the sensor
GAZEBO	sensorSensor	Car + obstacle	Separating distance	Check the data read by the sensor
	imuSensor	Car moving	Current speed	Check the data read by the sensor
	cmdFollowerCar	Speed command	X	Check that the car is moving when a speed command is received

FMEA (Failure Mode and Effects Analysis)

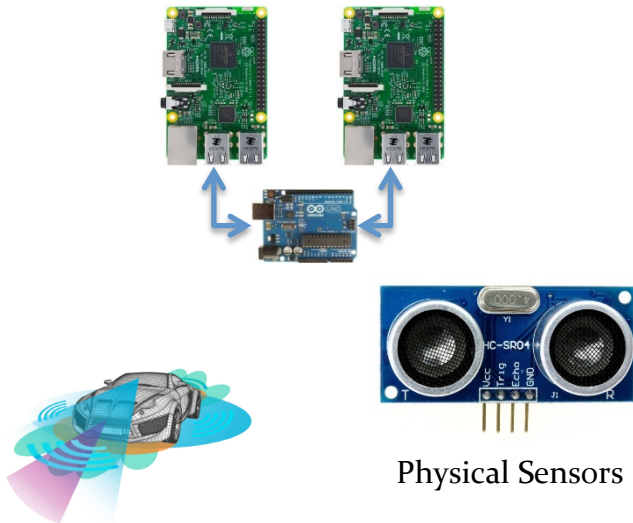
Entity	Failure mode	Risk	F	G	Safety mechanism
Raspberry	Crash	No speed command received by the car.	2	4	Passive replication
Simulator: Gazebo	Crash	No more car, the TJP does not exist anymore.	2	4	Alarm message + stop application
Roscore	Crash	No communication between nodes and topics anymore.	1	4	Alarm message + stop application
Virtual sensor: IMU	Crash	Set point distance no longer adaptive.	1	3	Alarm message + stop application
	Inconsistent data	Wrong set point distance.	3	3	Error message + Temporal redundancy
Virtual sensor: Laser	Crash	Collision with the front car.	1	4	Alarm message + Physical redundancy
	Inconsistent data	Wrong distance to the front car.	3	4	Error message + Physical redundancy
Real sensor: Ultrasonic	Crash	No detection of a close obstacle.	1	4	Alarm message + stop application
	Inconsistent data	Emergency braking wrongly activated.	3	4	Temporal redundancy

Frequency	Gravity			
	4	3	2	1
4	ASIL D	ASIL C		
3		ASIL B	ASIL A	
2				
1				QM

Automotive Safety Integrity Level – ASIL

Prototyping Mock-up

- **Simulator (PC)**
 - Cars
 - Virtual sensors
- **Physical platform**
 - Arduino Uno
 - Raspberry Pi 3
 - Real ultrasonic sensor










Physical Sensors



Conclusion

Summary

SoC	ROS nodes, component mapping to nodes	
D4A	Componentized FT design patterns Protocol-Before-Proceed-After	
Nodes Mngmnt	Unix system calls and ROS commands	 
Dynamic Binding	ROS services, ports, topics Additional logic to create ports and topics	  
Experimental Platform	Mock-up for validation, Hardware support, Executive support	