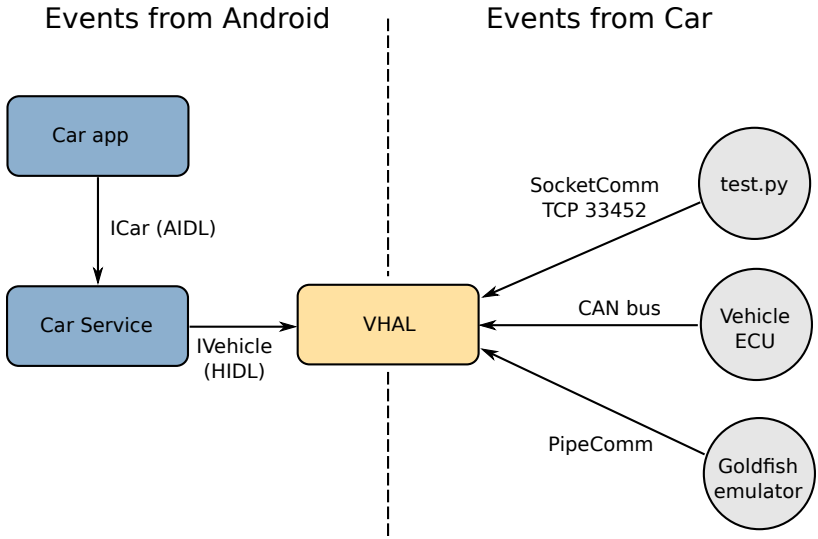
VHAL SocketComm

The problem

- How to test car apps without hardware (no car)?

The Vehicle HAL



Vehicle Properties

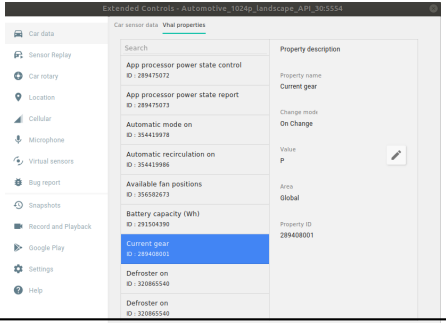
- Information collected from vehicle bus is represented in **Vehicle Properties**
- Examples
  - `PERF_VEHICLE_SPEED`: Speed of the vehicle in metres per second, floating point, read only
  - `GEAR_SELECTION`: gear selected by the driver, integer, read only
- System properties are defined in HIDL in file
  `hardware/interfaces/automotive/vehicle/2.0/types.hal`
- Vendor can extend with properties in VENDOR group

IVehicle

- **IVehicle** connects the car service to the VHAL
- Functions defined in
  `hardware/interfaces/automotive/vehicle/2.0/IVehicle.hal`
  - `get(VehiclePropValue)`
  - `set(VehiclePropValue)`
  - `subscribe(IVehicleCallback, SubscribeOptions)`
  - `unsubscribe(IVehicleCallback, propId)`
  - `getAllPropConfigs()`
  - `getPropConfigs(props)`

SocketComm and PipeComm

- SocketComm and PipeComm are part of the default VHAL daemon
- Both allow vehicle property reads and writes to be injected **as if they came from the car**
  - SocketComm: via TCP socket, port 33452
  - PipeComm: qemu pipe, used in Goldfish emulator:



VHAL and SocketComm

The code for the VHAL daemon is in
`hardware/interfaces/automotive/vehicle/2.0/default`

- Starts with `VehicleService` (`VehicleService.cpp`)
- Creates `VehicleEmulator` (`impl/vhal_v2_0/VehicleEmulator.cpp`)
- Which creates SocketComm (`impl/vhal_v2_0/SocketComm.cpp`)
  - also PipeComm (`impl/vhal_v2_0/PipeComm.cpp`)
- SocketComm listens on port 33452 for messages and injects them into the VHAL
- Message format is protobuf, auto generated from HIDL in IVehicle.hal and types.hal

SocketComm clients

- Usually written in Python
- Example code in `packages/services/Car/tools/emulator`, including:
  - `vhal_emulator.py`: class to connect to VHAL via SocketComm
  - `VehicleHalProto_pb2.py`: message interface to VHAL, using proto bufs
  - `diagnostic_builder.py, diagnostic_injector.py`: OBD2 diagnostics
  - `gui.py`: demo UI using QT4 (change gear; set indicators to hazard, turn left, turn right)

vhal_emulator.py

- Contains Vhal class which sends and receives messages to VHAL, using ADB port forwarding

Send a getConfig command for property `prop`

```
getConfig(self, prop)
```

Send a getProperty command for `prop`, `area_id`

```
getProperty(self, prop, area_id)
```

Send a setProperty command for `prop`, `area_id`, `value`

```
setProperty(self, prop, area_id, value, status=VehicleHalProto_pb2.AVAILABLE)
```

Receive a message

```
rxMsg(self)
```

vhal_consts_2_0.py

- Contains VHAL constants, auto generated from types.hal
- Example:

```
VEHICLEPROPERTY_PERF_VEHICLE_SPEED = 0x11600207
VEHICLEPROPERTY_GEAR_SELECTION = 0x11400400
```

Generated by `vhal_const_generate.py`

Example: read a property

read-prop-example.py

```
#!/usr/bin/env python

import vhal_consts_2_0 as c
from vhal_emulator import Vhal

if __name__ == '__main__':
    v = Vhal(c.vhal_types_2_0)
    v.getProperty(c.VEHICLEPROPERTY_ENV_OUTSIDE_TEMPERATURE, c.VEHICLEAREA_GLOBAL)
    reply = v.rxMsg()
    print(reply)
```

Testing

```
$ ./read-prop-example.py
Connecting local port 33005 to remote port 33452 on default device
msg_type: GET_PROPERTY_RESP
status: RESULT_OK
value {
  prop: 291505923
  value_type: 6291456
  timestamp: 18211762823719
  area_id: 0
  float_values: 25.0
  status: AVAILABLE
}
```

Example: write a property

write-prop-example.py

```
#!/usr/bin/env python

import vhal_consts_2_0 as c
from vhal_emulator import Vhal

if __name__ == '__main__':
    v = Vhal(c.vhal_types_2_0)
    # v.setProperty(c.VEHICLEPROPERTY_ENV_OUTSIDE_TEMPERATURE, c.VEHICLEAREA_GLOBAL, 42)
    v.setProperty(c.VEHICLEPROPERTY_ENV_OUTSIDE_TEMPERATURE, 0 , 42)
    # Get the response message to setProperty()
    reply = v.rxMsg()
    print(reply)
```

Testing

```
$ ./write-prop-example.py
Connecting local port 43211 to remote port 33452 on default device
msg_type: SET_PROPERTY_ASYNC
status: RESULT_OK
value {
  prop: 291505923
  value_type: 6291456
  timestamp: 28062262107
  area_id: 0
  float_values: 42.0
  status: AVAILABLE
}
```

Warning: there is a bug which causes carwatchdogd to timeout and reboot when you run this code. I'm working on it ...

gui.py demo

Quick start guide to setting up and running gui.py:

```
$ sudo apt-get install python-qt4
$ cd packages/services/Car/tools/emulator
$ ./gui.py
```

