Date: 10 May 2022

Location:Bangalore

rmunibyr@qualcomm

CAN Bus Virtualization

Radvajesh M

Sr Staff Engg

Qualcomm

Agenda

- CAN Introduction
- Typical CAN Deployment on Linux
- Need for Virtualization
- Choice for VirtIO CAN
- Early View of Virto CAN
- Early View of Virto CAN with Auto SAR

CAN Introduction

A Controller Area Network (CAN bus):

Is a robust vehicle bus standard designed to allow microcontrollers and devices to communicate with each other's applications without a host computer. It is a message-based protocol.

In 1983 Bosch published several versions of the CAN specification and the latest is CAN 2.0 published in 1991. Bosch is still active in extending the CAN standards. In 2012, Bosch released CAN FD 1.0 or CAN with Flexible Data-Rate. These standards are freely available from Bosch along with other specifications and white papers

In 1993, the International Organization for Standardization (ISO) released the CAN standard ISO 11898, The physical layer standards ISO 11898-2 and ISO 11898-3 are not part of the Bosch CAN 2.0 specification. These standards may be purchased from the ISO.

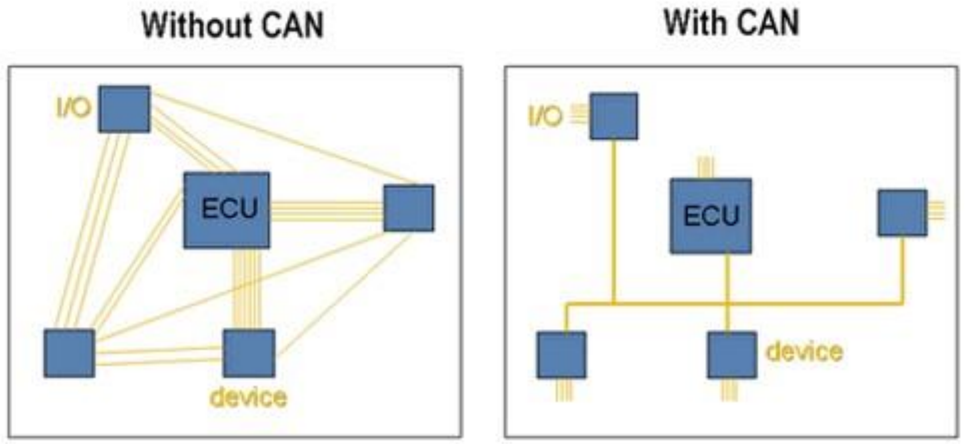
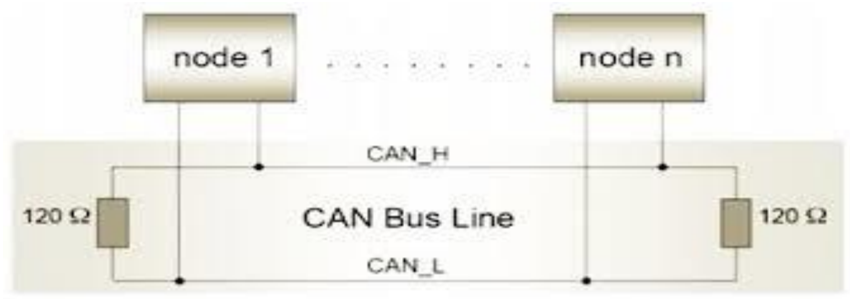
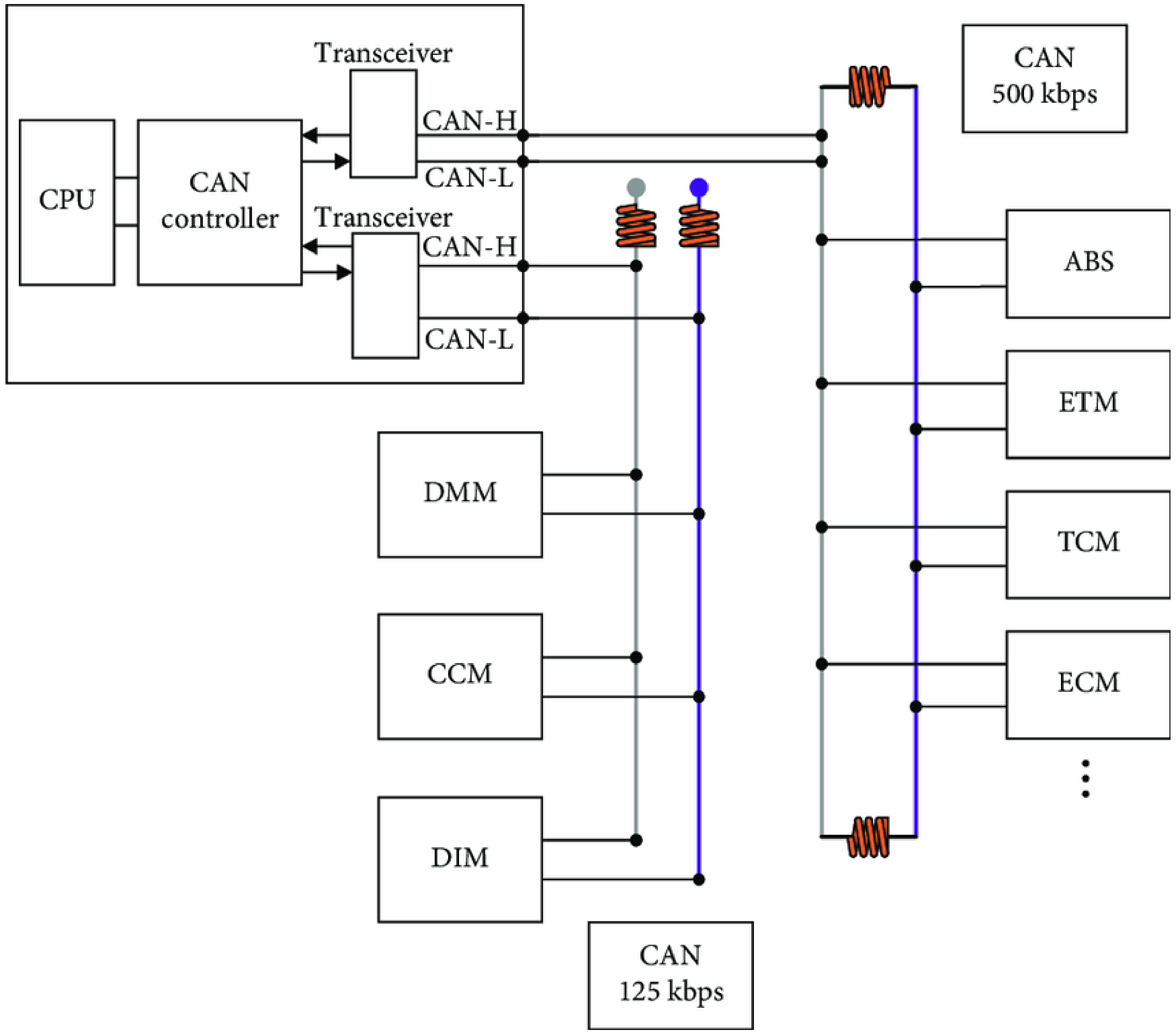


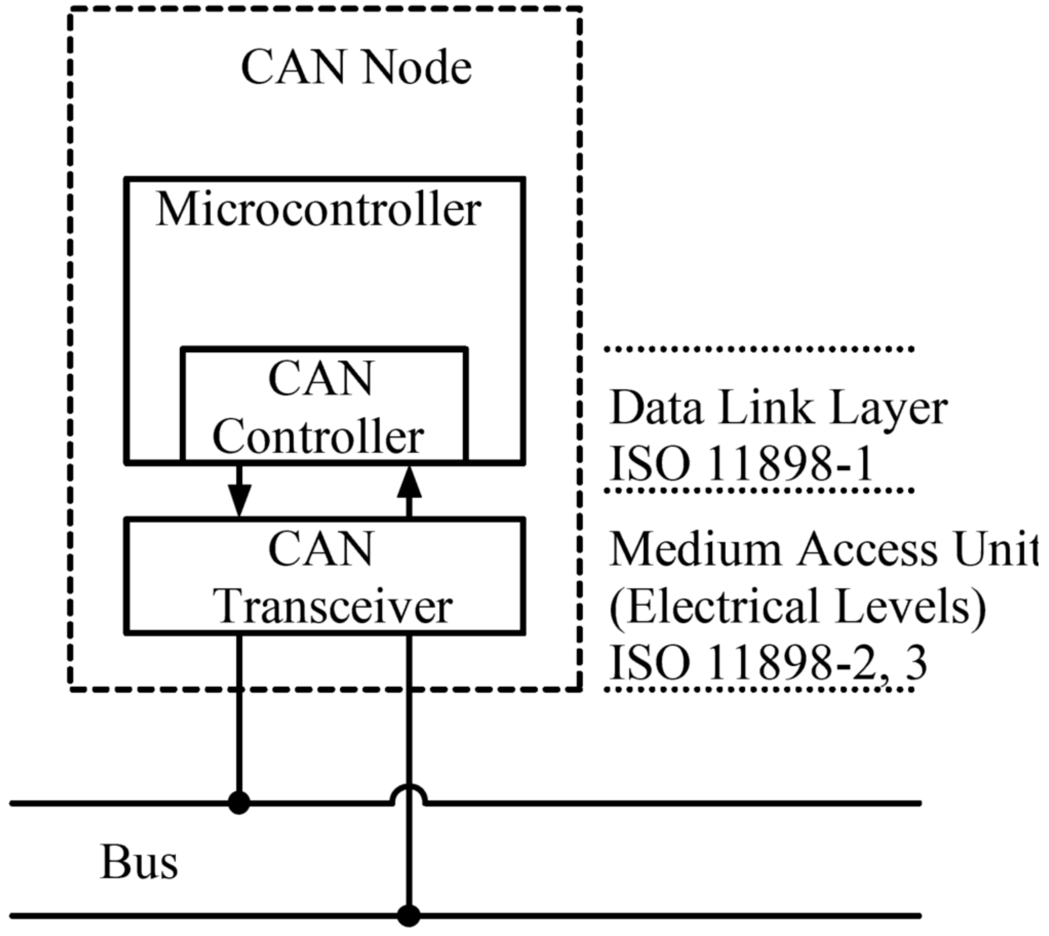
Image Ref:Image ref: https://www.ni.com/en-in/innovations/white-papers/06/controller-area-network--can--overview.html

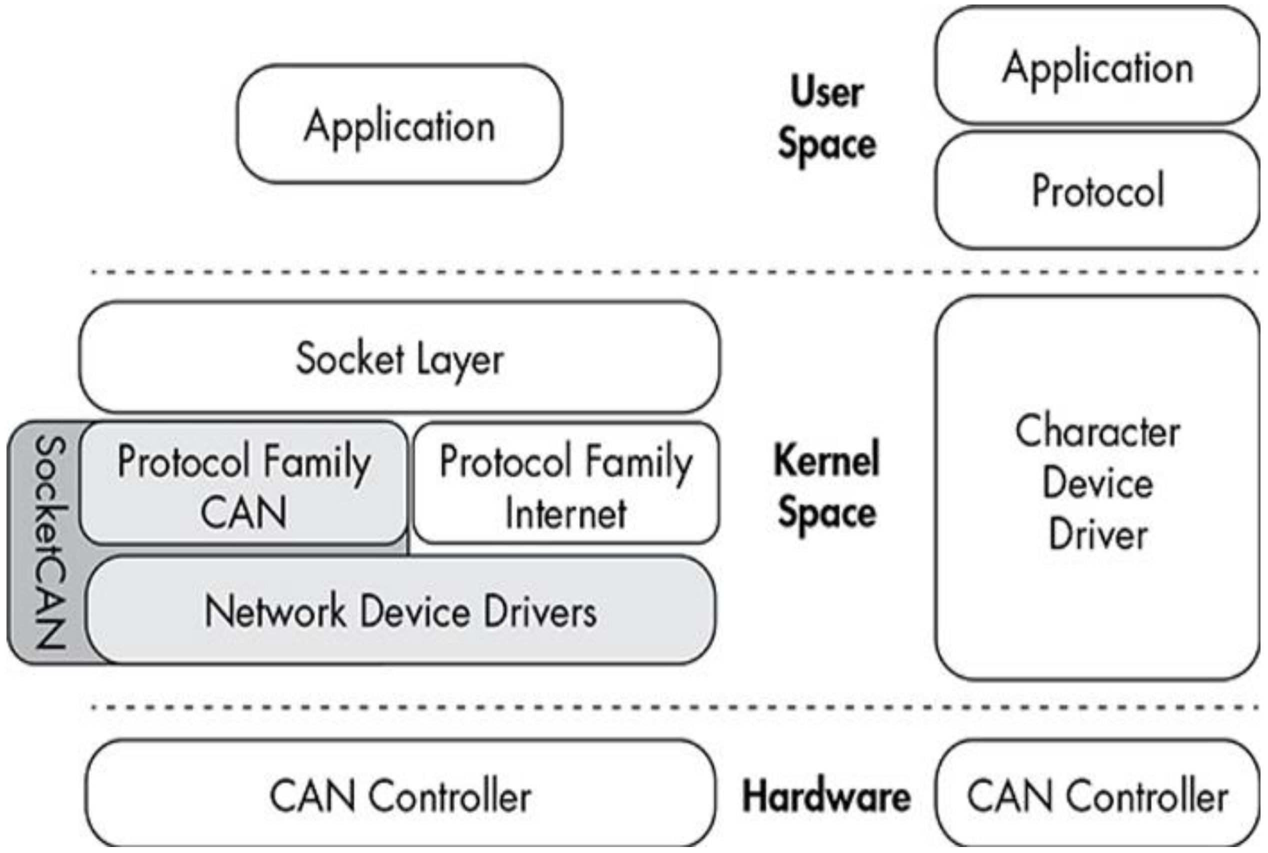
CAN Bus working Introduction





Typical CAN Deployment


CAN Node
Microcontroller
CAN Controller
CAN Transceiver
Data Link Layer ISO 11898-1
Medium Access Unit (Electrical Levels) ISO 11898-2, 3
Bus



Source: https://medium.com/@xesey/listening-vehicle-communication-with-socket-can-8545e0241fa

CAN Features

- Support Classical CAN TX/RX
- Standard ID:11 ID, 5 CTRL, 8 byte Data)
- Extended ID (29 ID, 8 CTRL, 8 byte Data)
- Support for CAN FD (64 bytes data)
- Data BitRate: 2Mbps
- Arbitration configurable upto 1Mbps
- Support to Add Filters for RX
- Support for Early CAN Buffering
- Support for Time Stamping CAN Frames
- Error Handling

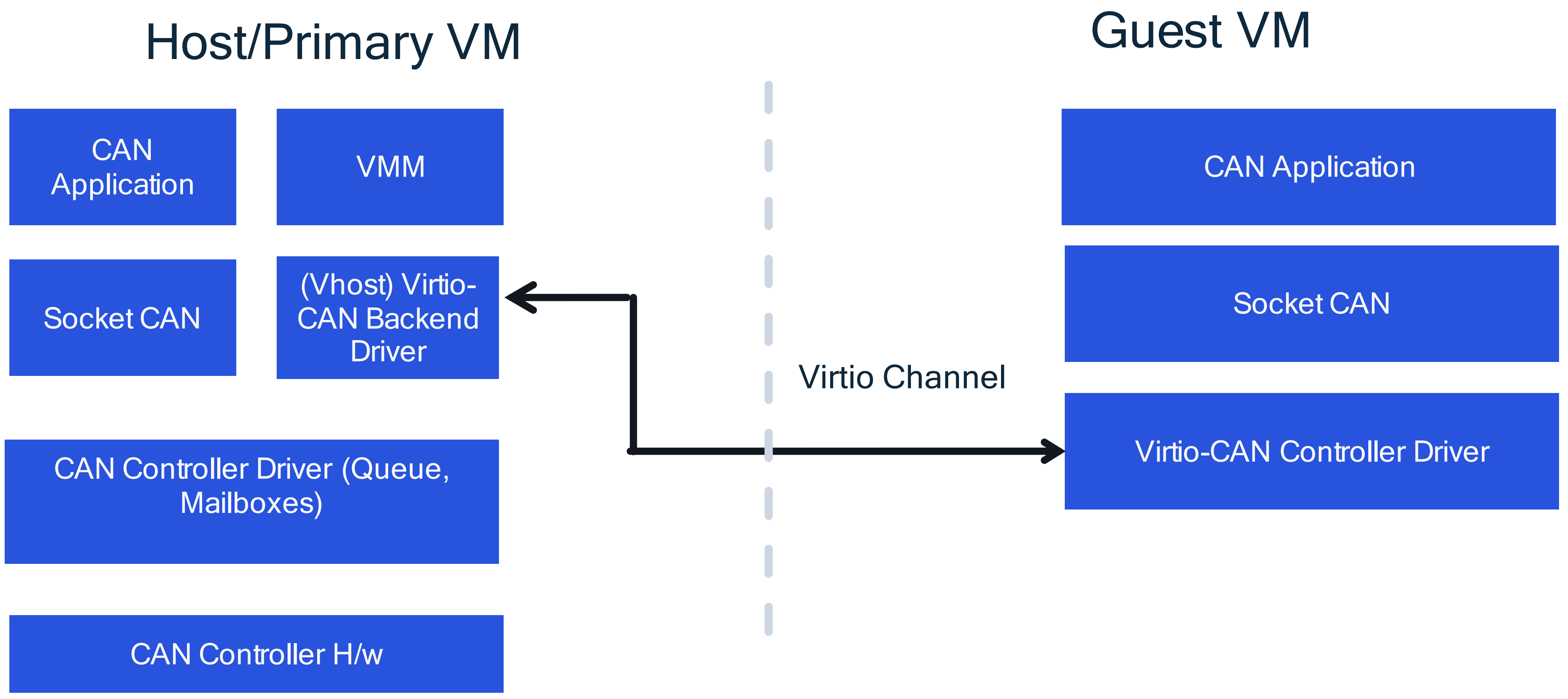
Need for Virtualization

- Virtualize CAN bus, so that multiple VMs can participate in CAN bus transactions as if each VM is directly connected to physical CAN bus

- Chief motivation - Reduce electronics/cost, simplify

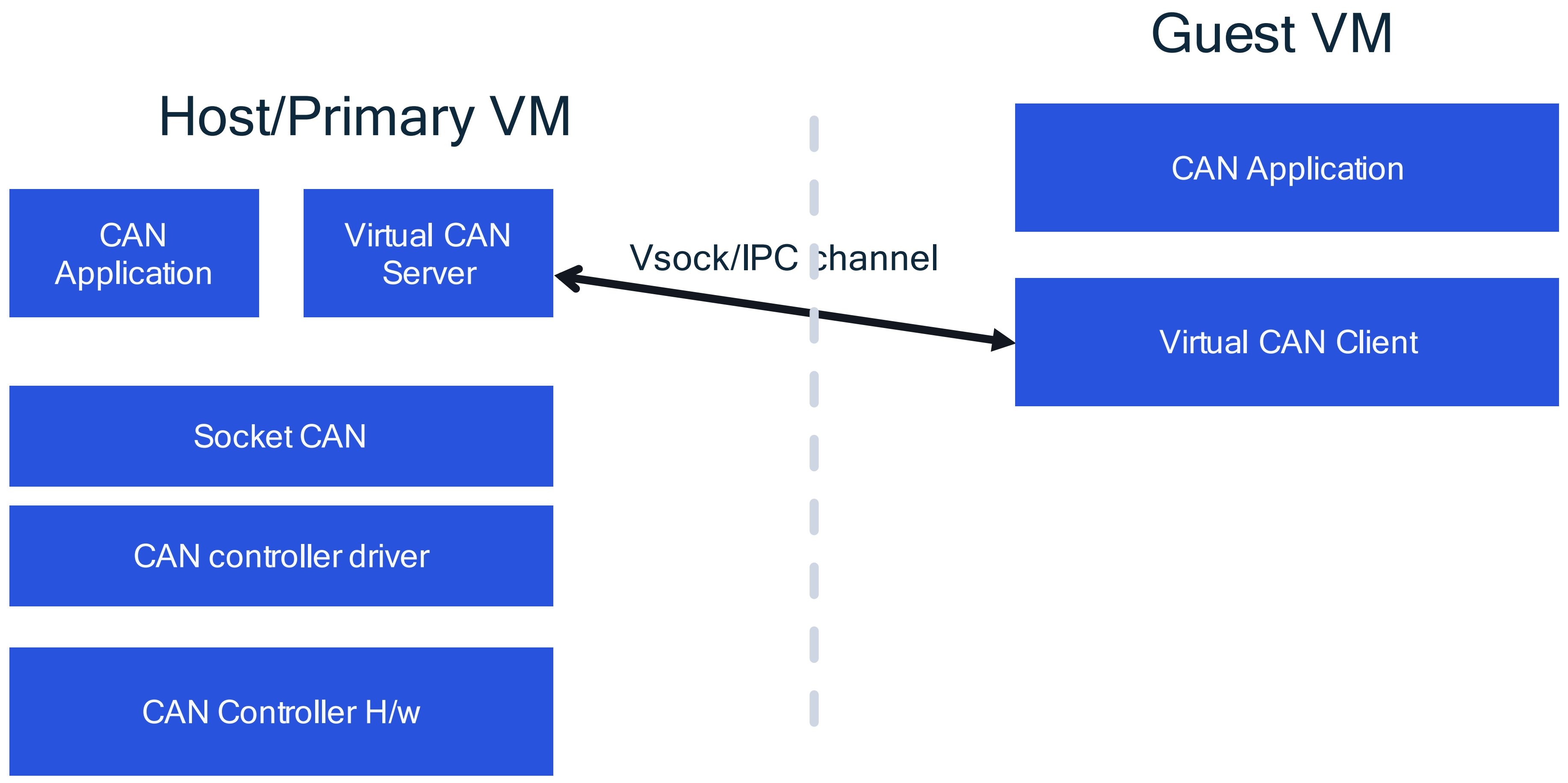
- Challenges – CAN bus arbitration, broadcast, performance/latency requirement

CAN bus virtualization - Option 1



- CAN application in guest VM to use standard CAN API provided by guest OS
- Virtio-CAN specification needs to be developed in this scenario

CAN bus virtualization - Option 2



CAN application in guest VM to use custom IPC API to communicate with a Virtual CAN server

Virtio-CAN specification NOT needed in this case. Standardize the IPC communication?

Virtio CAN – Prior Work

- RFC posting on virtio-dev by Open synergy
  - Does not support some features like filtering, timestamping, …
- Xvisor implementation
  - Virtio-backend in hypervisor

Source: https://hal.archives-ouvertes.fr/hal-01291895/document

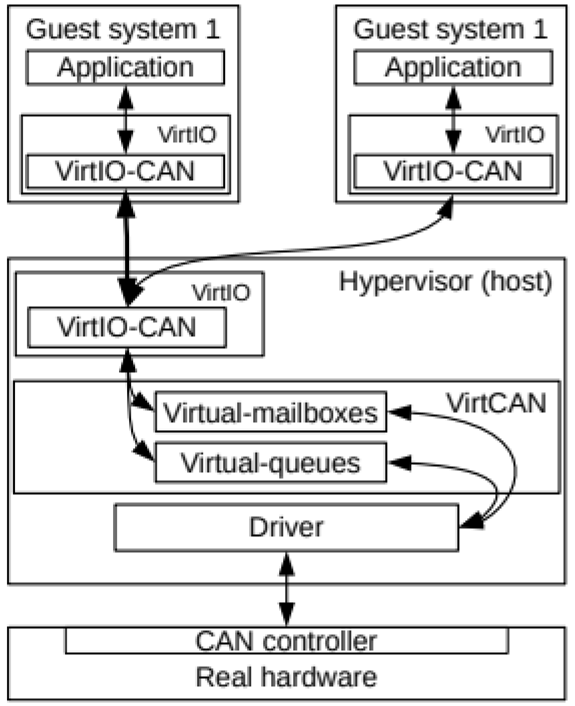
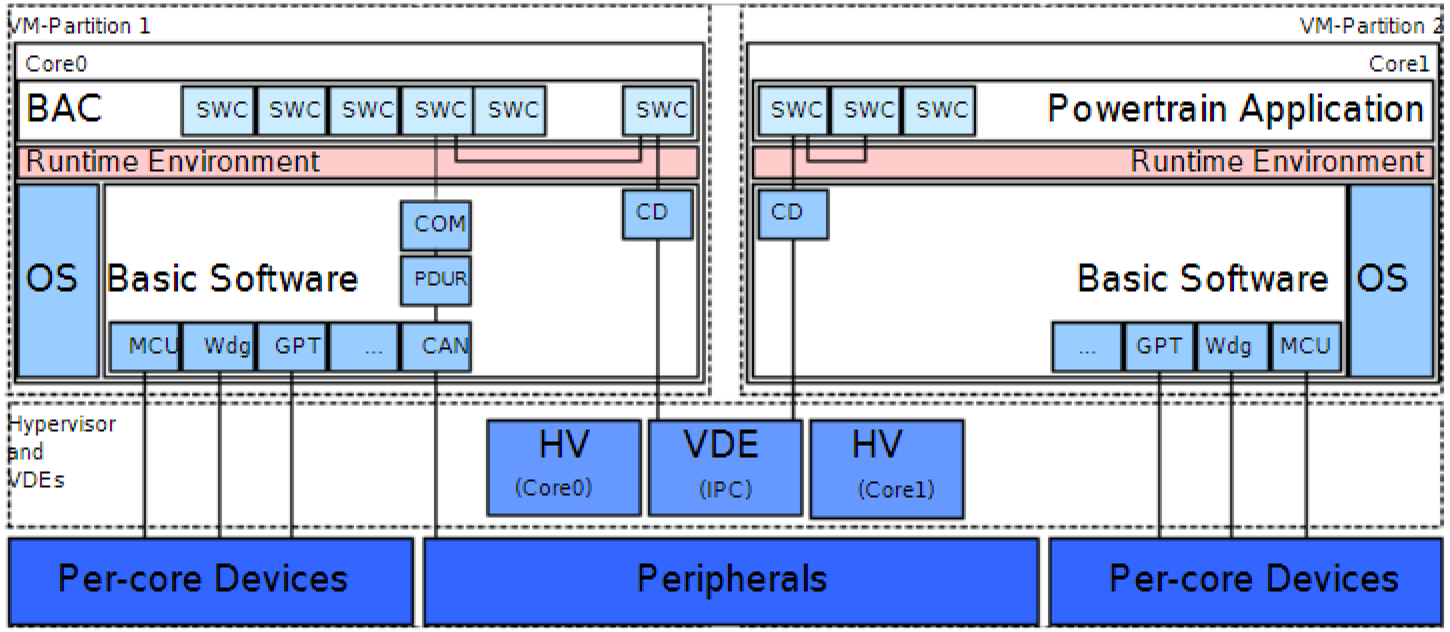


Fig. 9. System design

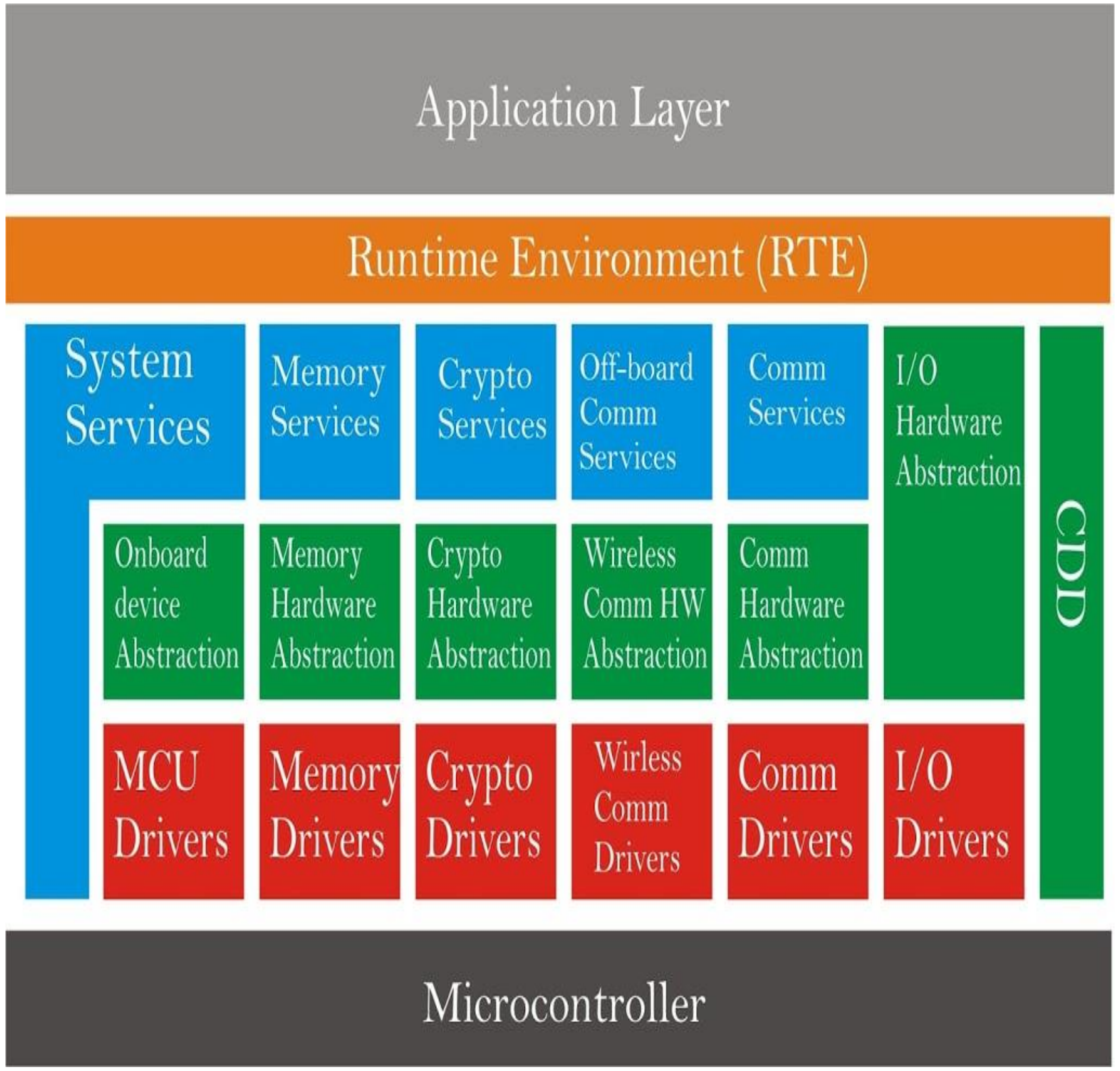
BACK UP

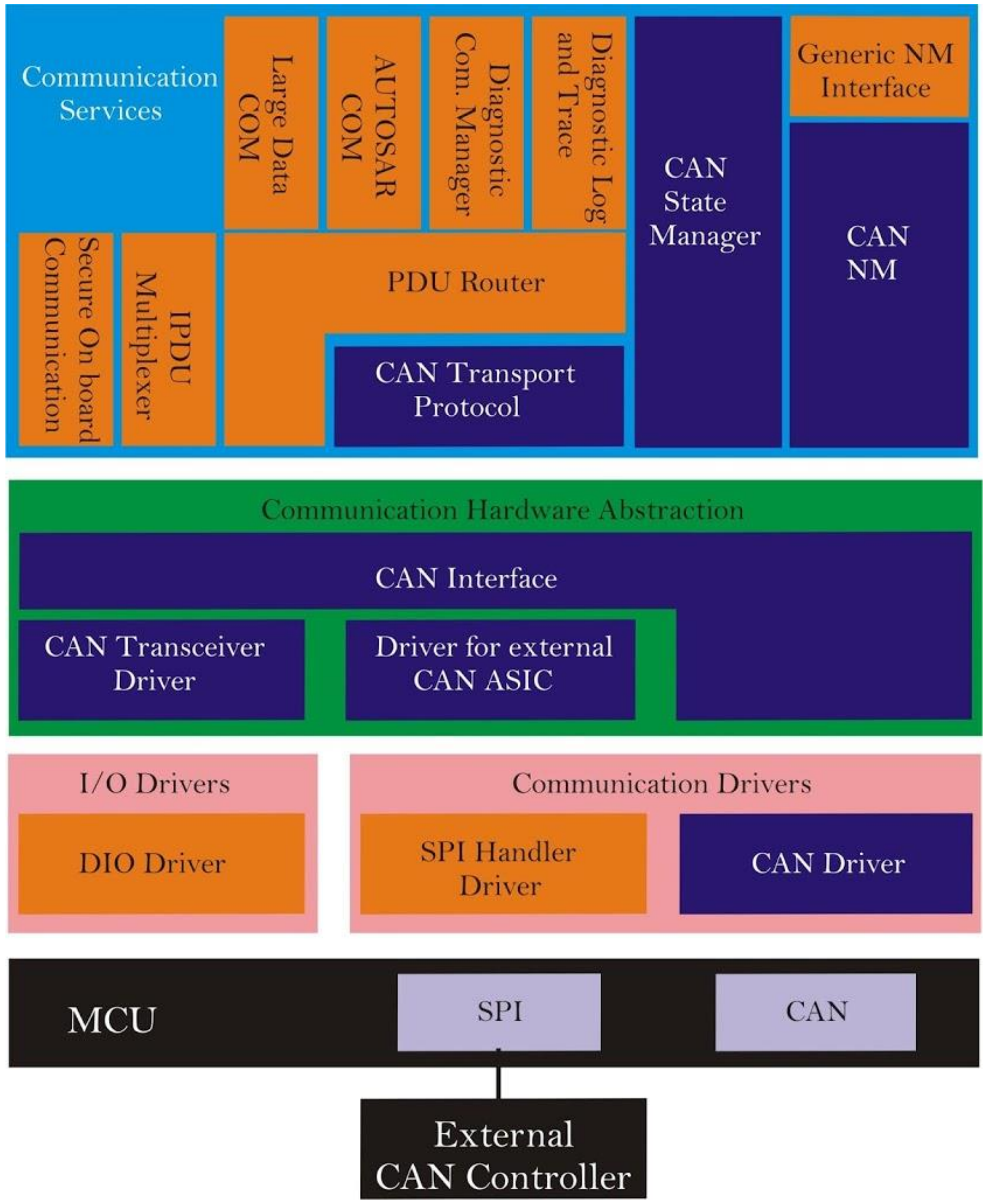
Early View of VirtIO CAN on AutoSar



Source: https://www.researchgate.net/publication/271417105_An_Embedded_Hypervisor_for_Safety-Relevant_Automotive_EE-Systems

AutoSAR CAN Stack

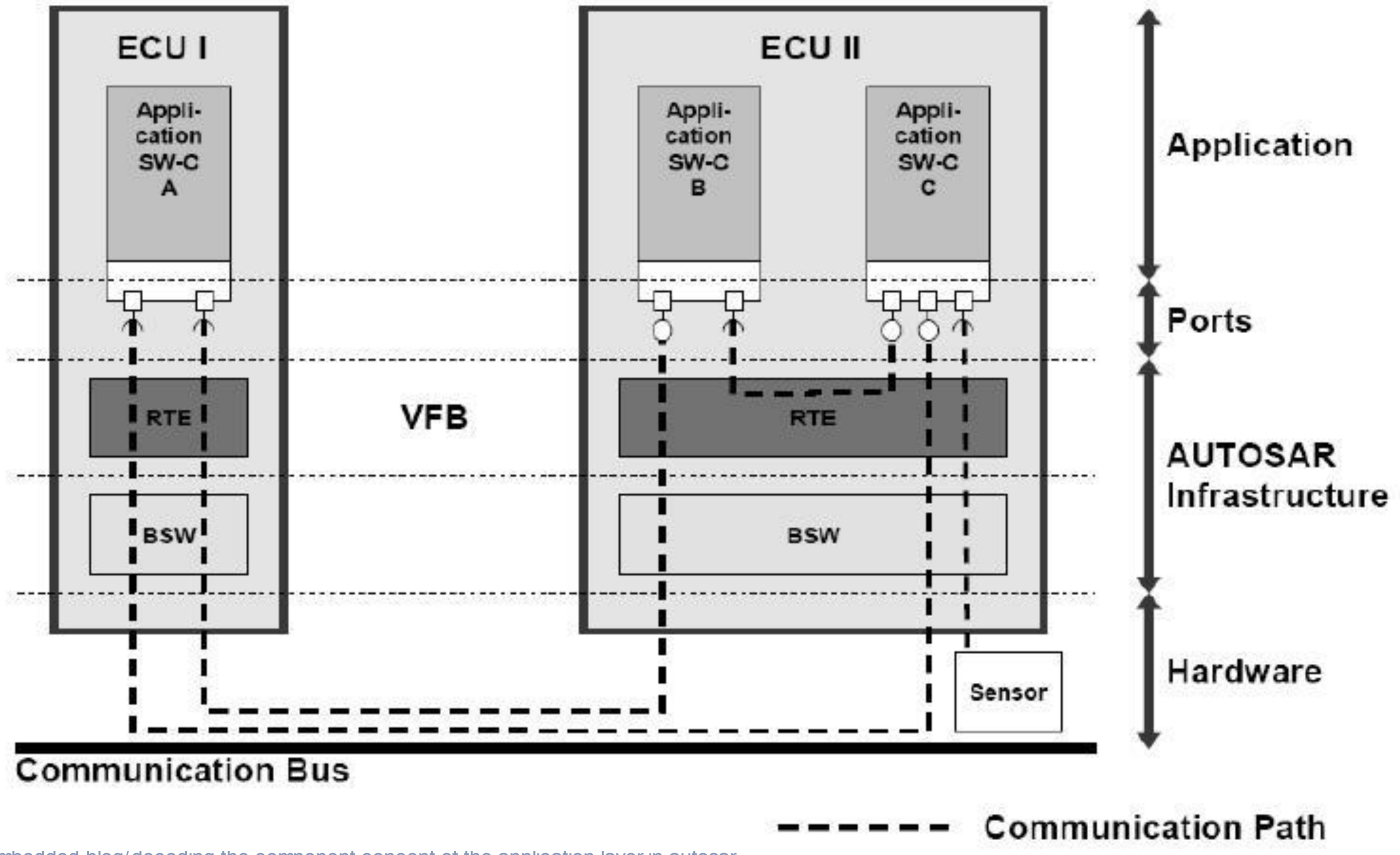




Source: https://autosartutorials.com/tag/autosar-can-stack/

AUTOSAR Virtual Bus

Intra- and inter-ECU Communication

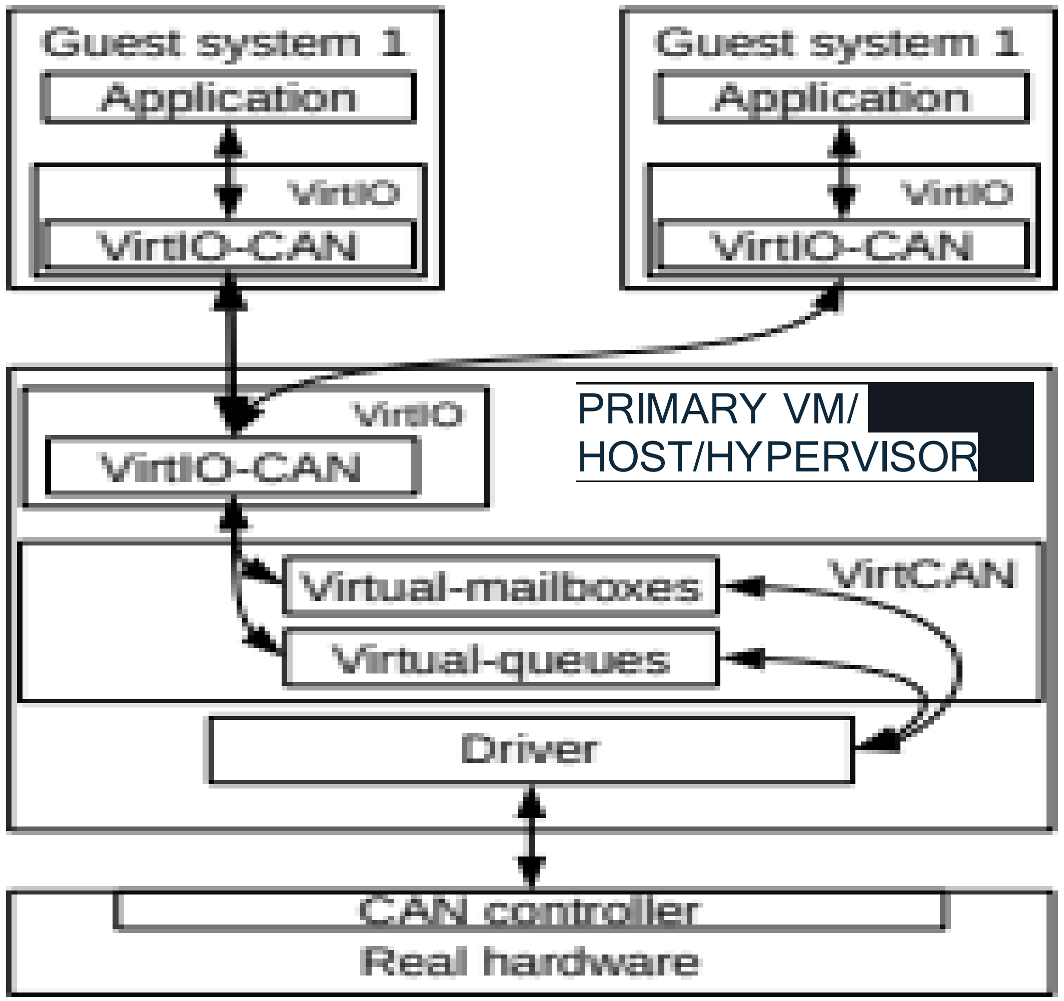


MW hides the distribution and the characteristics of the HW platform

Compliance: SW-C must only call entry points in the RTE

Source: https://www.embitel.com/blog/embedded-blog/decoding-the-component-concept-of-the-application-layer-in-autosar

Early View of VirtIO CAN



Source: https://hal.archives-ouvertes.fr/hal-01291895/document

Choice for VirtIO CAN

As CAN resources are shared, direct hardware access, namely passthrough, cannot be used. This would give all guests the complete access to the same hardware and registers in particular. A guest could then overwrite the controller configuration or the data set by another guest. In addition, it may perform conflicting operations that cause errors. In the worst case, it may render the whole controller or system unsusable. Thus, direct hardware access can only be used for exclusive hardware access from one system, either the host or one guest.

The guest high level (userland) certified softwares must remain the same. Thus, the interface for the CAN must not change, i.e. it has to present a SocketCAN compatible driver for LINUX guests

We choose to use VirtIO for CAN interface virtualization for three reasons. On the one side, VirtIO defines a clear and flexible API with well defined and optimized transport abstractions, that fit our needs. On the other side, VirtIO API is becoming the virtualization standard for host-guest communication interface