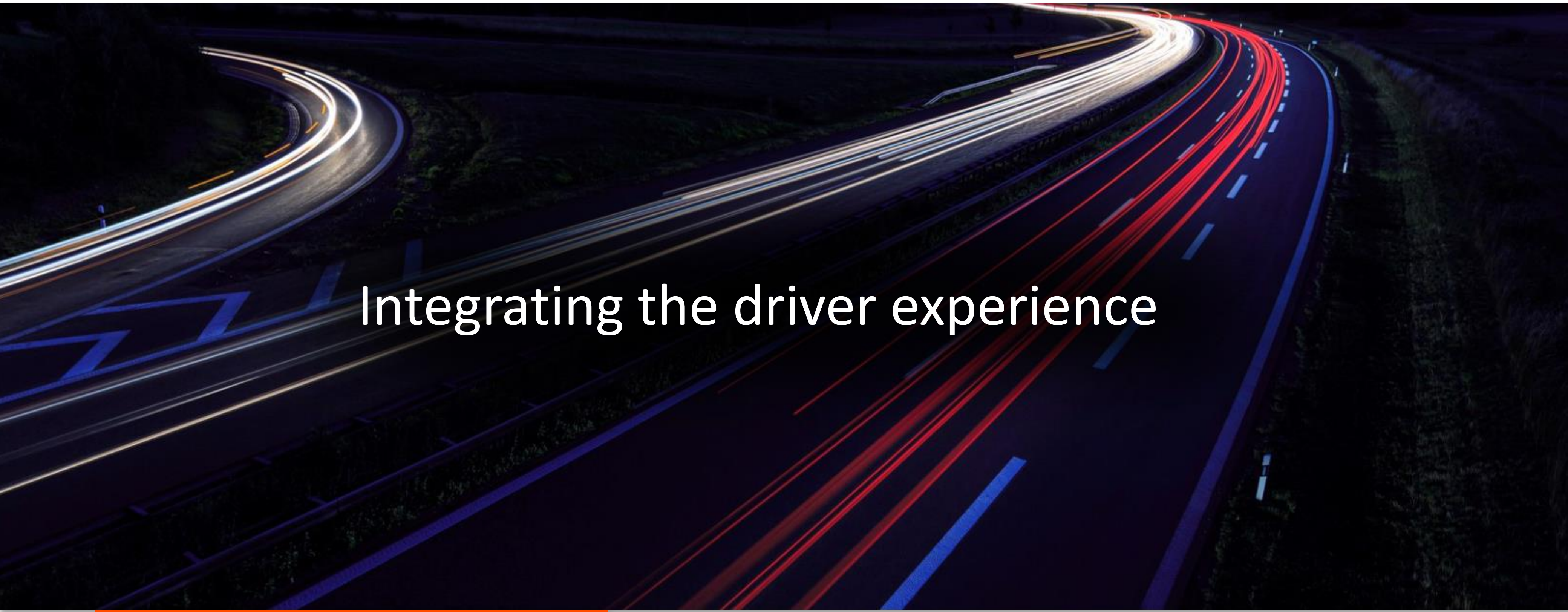
OPENSYNERGY



Paving the Path to Standardization of Virtualization

2018-06-20 Automotive Linux Summit (Tokio)

Dr. Ralph Sasse, Lead Solution Engineer

History, Background and Motivation

Hardware Sharing in **Server Farms** is unthinkable without virtualization support.

- IBM's CP-40 system, in mass production since 1967

Starting in 2005, CPU vendors have added **hardware virtualization** assistance to their products

Virtualization is nowadays commonly used on **Personal Computer** Desktop Machines.

Hypervisor are getting momentum in embedded **automotive** electronic control units on feature rich and powerful application processors.

- OpenSynergy’s Telematics Connectivity Unit, in mass production since 2014

Virtualization systems can run multiple software systems with very different requirements independently of each other.

Role of a hypervisor in automotive

A hypervisor makes it possible to build *mixed-criticality systems* and can integrate safely and securely:

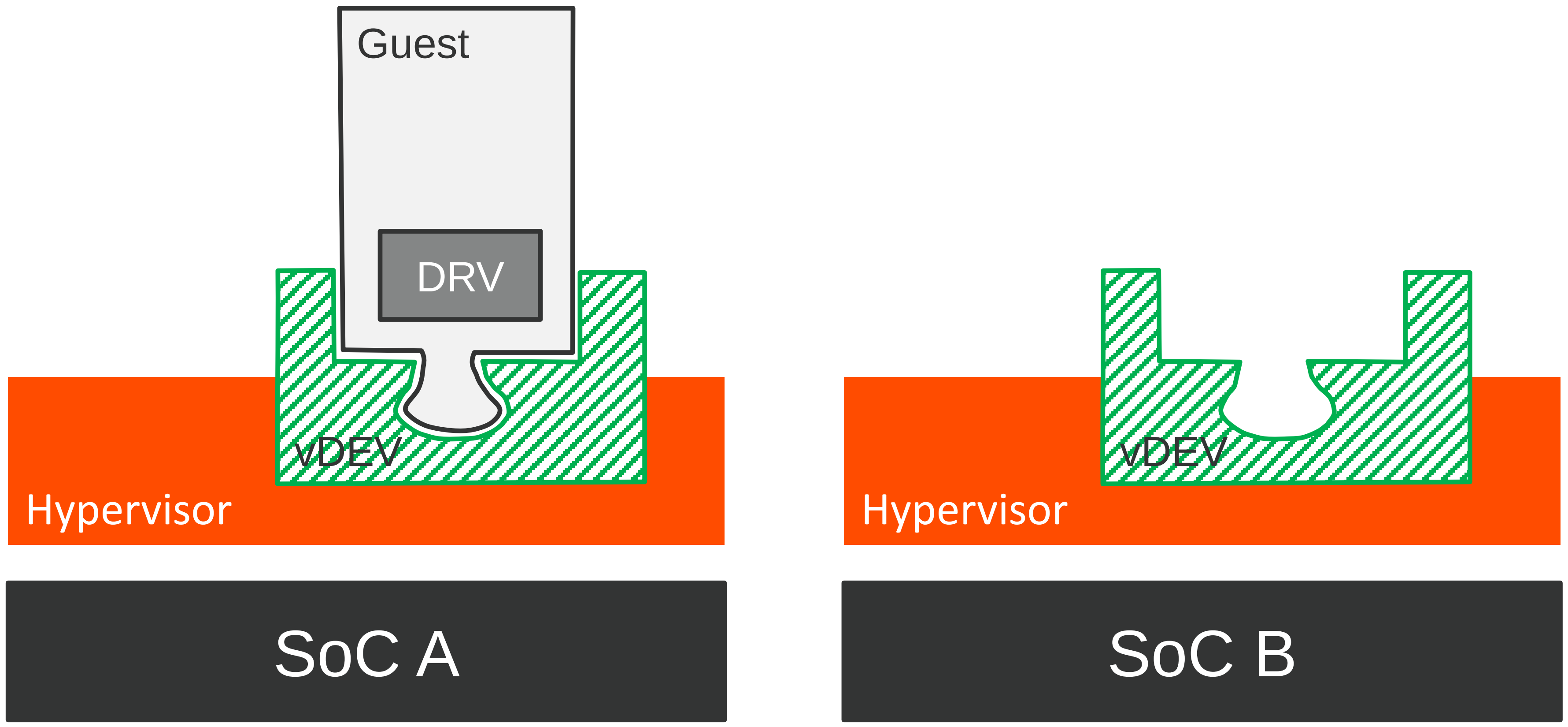
- Software that is developed according to different quality standards:
  - **Related to safety** (different ASIL levels, isolating safety from functionality)
  - **Related to security** (different levels of trust, isolating attacks)
  - **Related to reliability** (different levels of fault tolerance, isolating faults)
- Software that has different real-time (e.g. RTOS vs. generic OS) and boot-time requirements

A hypervisor:

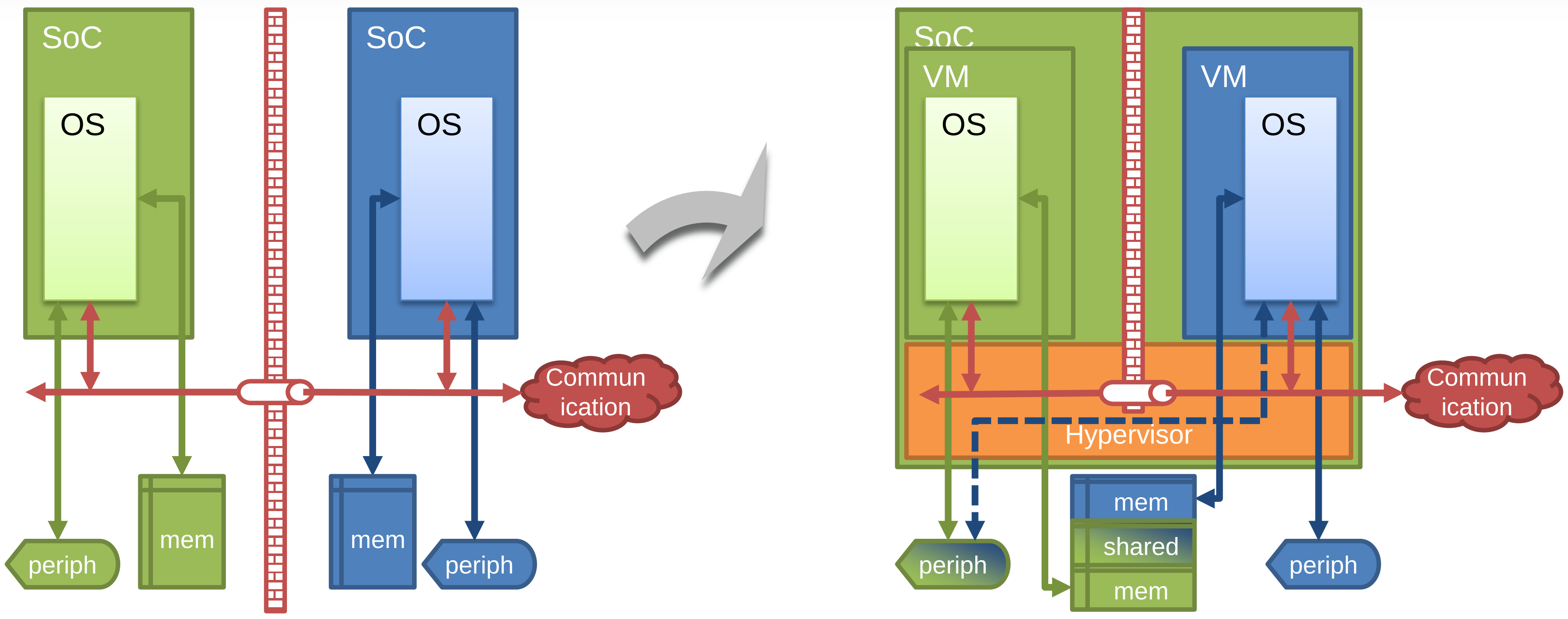
- Enables the use of optimally suited operating systems and frameworks (Linux, Android, AUTOSAR, RTOS)
- Isolates faults (safety, reliability) and attacks (security) and therefore minimizes Qualification/Certification effort
- supports ASIL and MILS decomposition
- Enables modular development and software updates

Vision „Virtual Platform”

A **Virtual Platform** would allow the development of virtual machine guests that could be moved among different hypervisor systems and/or HW platforms **without further modification**.



Concepts - Convergence

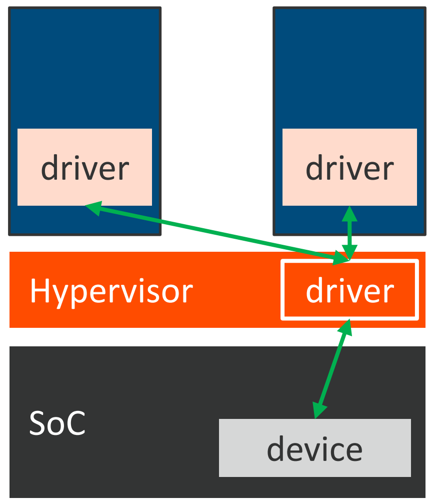


Target: Converge multiple Operating Systems on a single SoC

- Guaranteed independence of the individual Operating Systems by the Hypervisor (Isolation in Time and Space, ISO26262)
- Cooperation (via Communication) - with monitoring of conformity
- Additional option of cooperation (via **Sharing**) - with monitoring of conformity

Concepts - Device Virtualization in COQOS

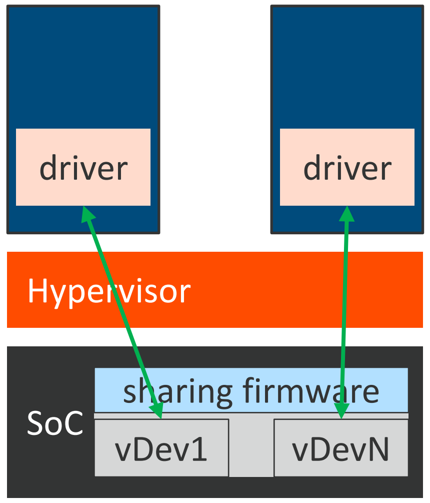
in Hypervisor



- Only used for UART (optionally)
- not recommended for other devices as the Hypervisor is minimalistic.

Example: UART

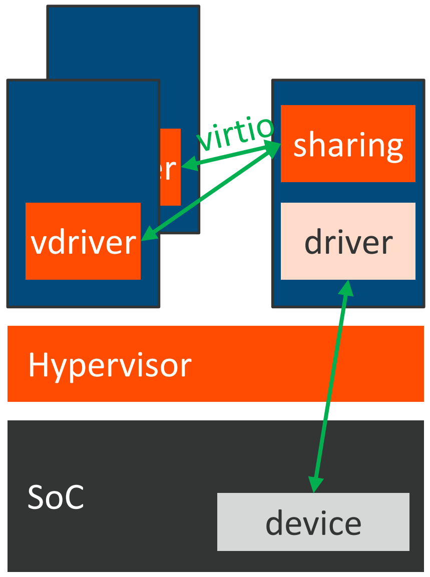
device with virtualization support



- COQOS supports this when the SoC hardware supports virtualized devices
- Recommended wherever the hardware supports it, as it tends to give the best performance and separation

Example: GPU on RCAR-H3

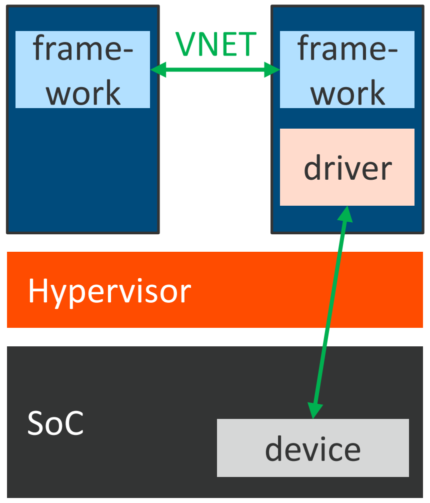
low-level client-server



- Single driver in VM that acts as "server"
- Driver-specific sharing logic is needed.
- Other VMs use "virtual driver".
- Compromise between performance and flexibility

Example: shared block device

distributed frameworks over VNET



- Allows reuse of existing frameworks for distributed applications in a virtualized environment over VNET.
- Supports complex sharing semantics at the cost of more overhead

Example: NFS, PULSE AUDIO

Low-Level Device Virtualization Technologies

OPENSYNERGY

	Description	Reusability	Platform independence	
Standard library virtualization (OpenGL, DRM, Android HAL …)	Implement hypervisor specific standard libraries	As long as the same hypervisor is used	As good as the vendor interface	😐
VIRTIO	Implement virtio based devices that follow either existing standards or specify new ones	virtio support is available in Linux, Android and many other operating systems	Builds upon the kernel-userspace interface of Linux and allows large flexibility because the devices themselves make no assumption about the hardware	🙂
Hypervisor vendor custom	Develop virtual devices optimized to be used with a particular hypervisor	As long as the same hypervisor is used	Implementation specific	☹

Trade-off between development effort, reusability, platform independence, availability and maturity

Introduction to VIRTIO

- **VIRTIO "De-Facto Standard For Virtual I/O Devices" [Russel 2008]**
- **Formally standardized since March 2016 (OASIS VIRTIO-v1.0)**
- **VIRTIO provides the transport layer and device models for many devices**
  - Block Storage, SCSI 🟢
  - Network 🟢
  - Console 🟢
  - crypto 🟢
  - GPU 🟡
  - Input (hid) 🟡
  - vsock 🟡
  - 9pfs (File Server) 🟡
  - Many more in development (vIOMMU, etc.) 🟡
- **For the Automotive domain there are still missing pieces**
  - Audio 🔴
  - Sensors 🔴
  - Media Acceleration (VPU, IPU, CODEC) 🔴
  - USB, CAN, Ethernet AVB 🔴

🟢	specified
🟡	implemented
🔴	missing

Legend for VIRTIO terms

Device refers to the implementation of the virtual/para-virtual device, also known as Backend or Server

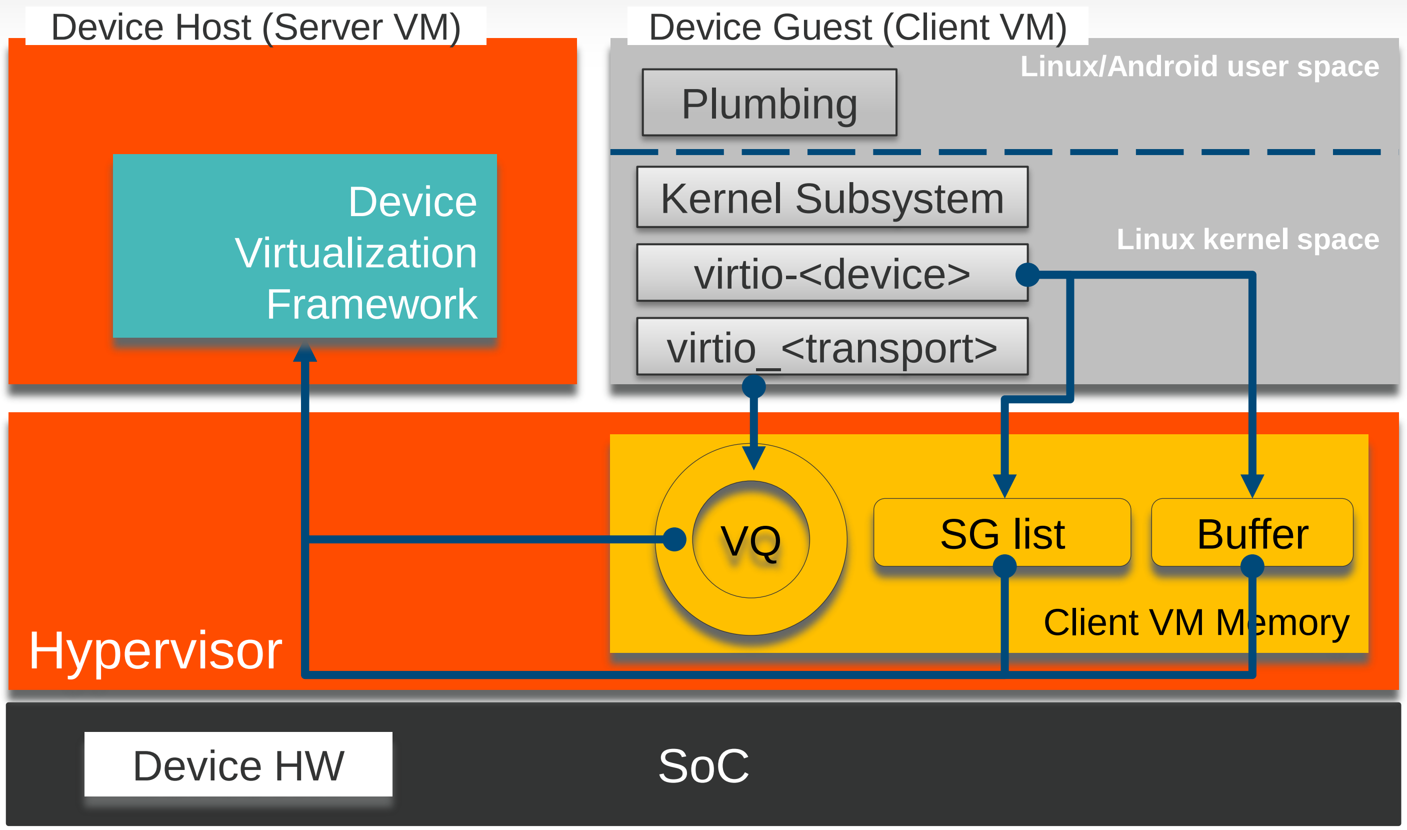
Driver refers to the guest driver, also known as Frontend or Client

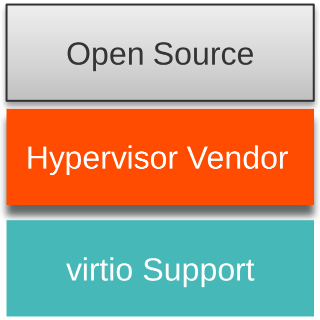
Device Host is the guest that provides the Device to other guests

Device Guest is the consumer of a Device

Guest is a partition or virtual machine

Virtualized device Architecture with VIRTIO





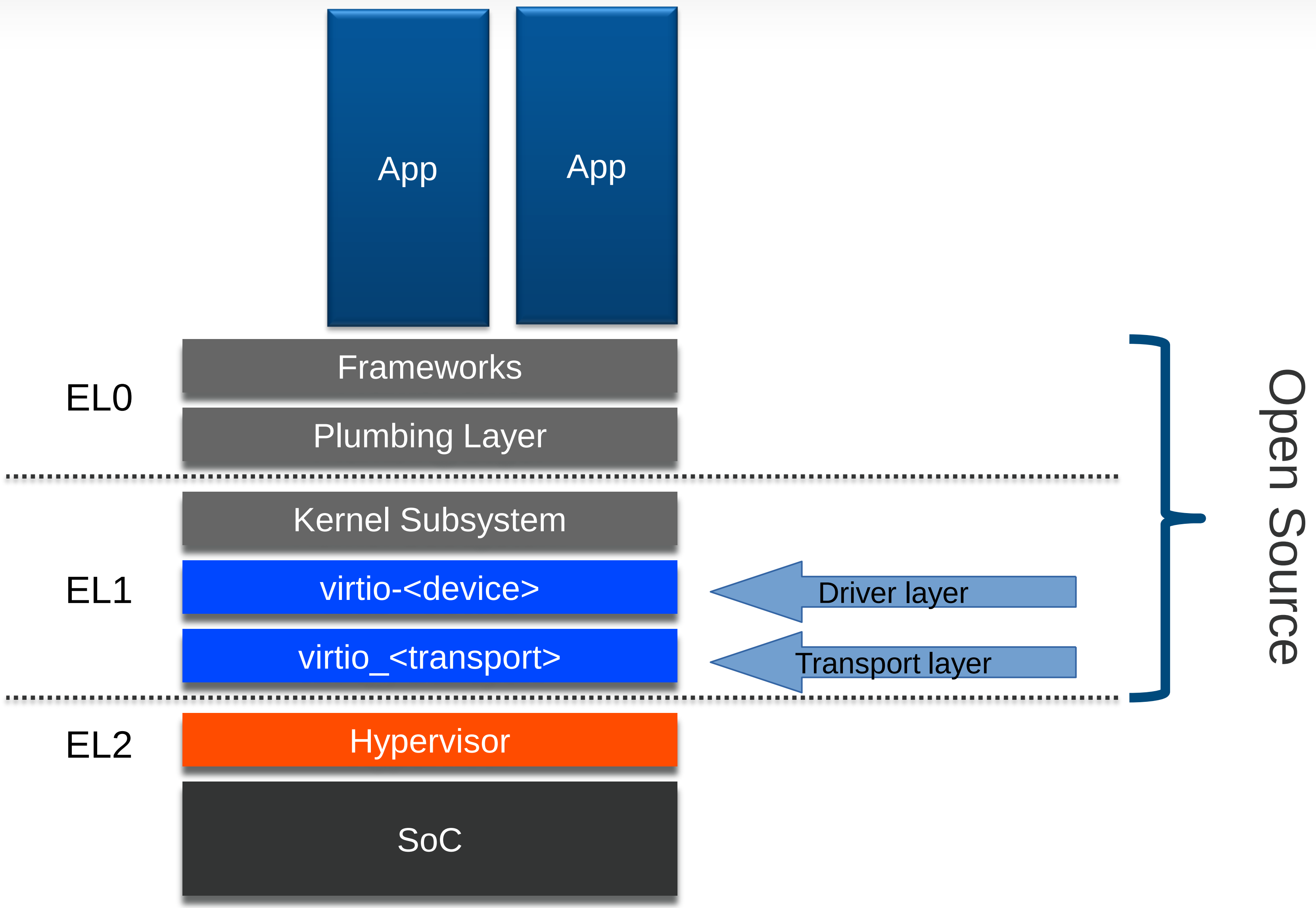
VQ=virt-queue
SG=Scatter Gather

Bulk data transport via DMA-like memory model

- Buffer allocations handled by „Driver“ part (client)
- Direct R/W access to allocated buffers in the „Device“ part (server)

Metadata transport via virt-queues (ring buffers, asynchronous pipeline)

VIRTIO driver Stack



Benefits of VIRTIO

- **Standardized**
- **Proven in Use**
  - Well tested device models
- **Established community**
  - IBM, Red Hat, Siemens, Huawei, Oracle, ARM, Intel
- **Efficient and performant**
- **Diverse operating system support**
  - Linux, BSD, Windows, UEFI
  - Driver maintenance done upstream
- **Supported by many VMMs and Clouds**
  - Qemu, kvm-tool
  - ARM Foundation model / Fast model
  - Google Compute Cloud, DigitalOcean, OHV

Benefits for the involved parties

Community

- Reuse in different domains (economy of scale)

Market

- More mature solutions
- More flexibility / choice

Customer

- Faster time to market
- Smaller price

Hypervisor vendors

- Less effort / maintenance

ToDo‘s for standardization

Required I/O devices shall be defined and agreed by the automotive industry and hypervisor vendors.

Neutral industry bodies act as forum between

- Hypervisor Vendors
- SW-Tier 1/OEM
- Hardware manufacturers

GENIVI

- Maintain and evolve automotive domain specific APIs and standards

AGL

- Provide collaboration with upstream kernel project and Linux Foundation

Linux Foundation

- Connect Automotive with Cloud + Enterprise Computing

All: Establish regular events for interoperability testing (plug feast) and standard steering

Summary and Outlook

The availability of powerful SoCs allows Convergence of multiple ECUs into a single ECU.

Convergence enables efficient device sharing.

Device sharing currently lacks standardization.

VIRTIO is ***THE*** candidate to harmonize the interface between guest and hypervisor.

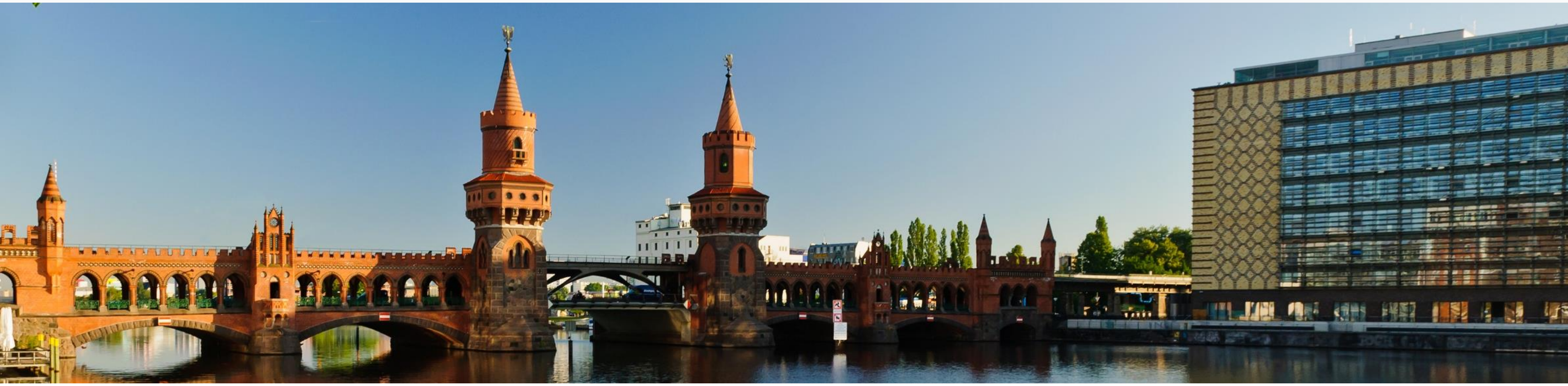
Neutral industry bodies (GENIVI, AGL, and Linux Foundation) should act as forum between

- Hypervisor Vendors
- SW-Tier 1/OEM
- Hardware manufacturers

Based on the result we can tackle the vision „Virtual Platform"

- VMs without modification

Questions? Comments?



Headquarter

Further Locations

Berlin	Utah	California	Munich
OpenSynergy GmbH	OpenSynergy, Inc. (USA)	OpenSynergy, Inc. (USA)	OpenSynergy GmbH
Rotherstraße 20 D-10245 Berlin Germany Phone +49 30 / 6098 5400	765 East 340 South Suite 106 American Fork, Utah 84003 USA	501 W. Broadway, Suite 832 San Diego, California 92101 USA Phone +1 619 962 1725	Starnberger Str. 22 D-82131 Gauting / Munich Germany Phone: + 49 89 / 8934 1333

E-Mail	info@opensynergy.com
Web	www.opensynergy.com