Virtual Open Systems

VOSYSVirtualNet: Low-latency Inter-world Network Channel for Mixed-Criticality Systems

2018-06-08

contact@virtualopensystems.com

www.virtualopensystems.com

About

- **Speaker: Julian VETTER**
  - Software Engineer at Virtual Open Systems (VOSYS) since 2017
  - Ph.D. from TU Berlin (SecT - Security in Telecommunications research group)
    - Thesis: "Strengthening System Security on the ARMv7 Processor Architecture with Hypervisor-based Security Mechanisms"
  - **Interests:** Operating Systems, Virtualization, Emulation, Embedded Devices and Software Security.

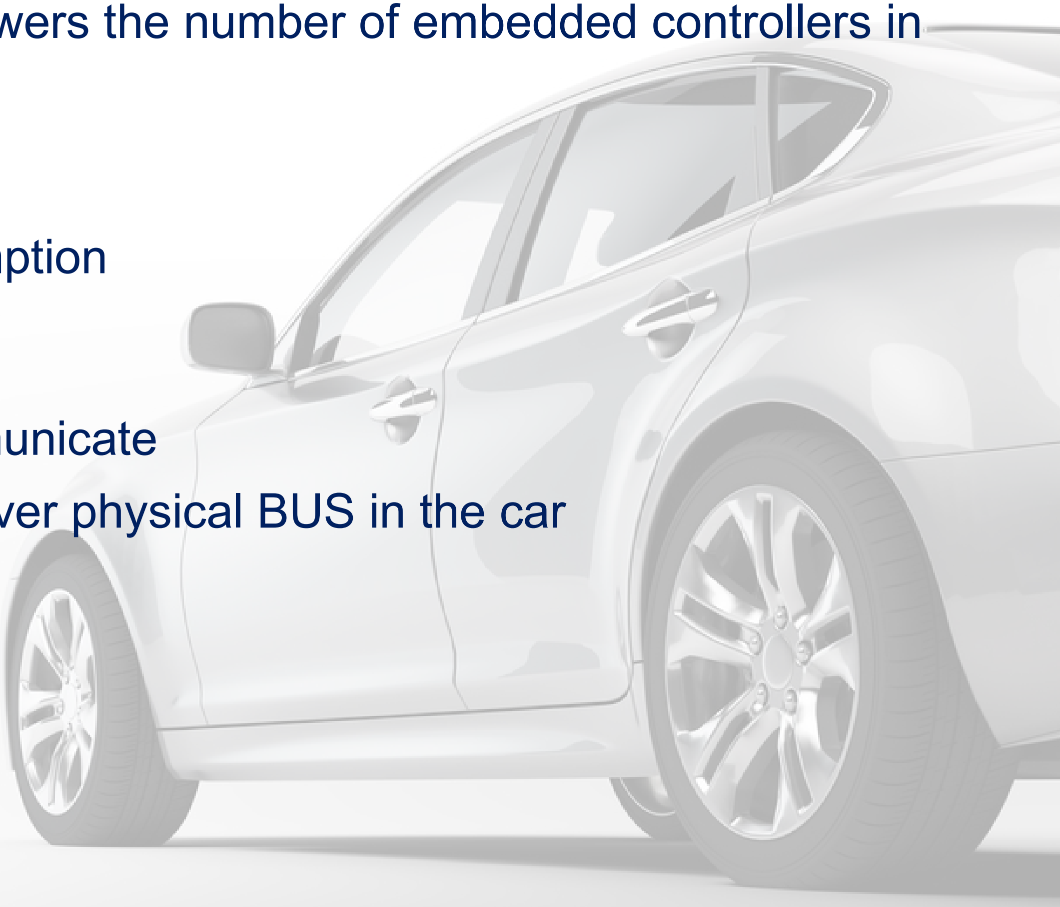
- **Company: Virtual Open Systems** is a high-tech software company active in open source virtualization solutions and custom services for complex mixed-criticality automotive, NFV networking infrastructures, consumer electronics, mobile devices and in general for embedded heterogeneous multicore systems around new generation processor architectures.

- **Funding: dRedBoX (http://www.dredbox.eu/)**
  - This work was funded through European Union’s Horizon 2020 research and innovation program, grant agreement No. 687632

Virtual Open Systems Confidential & Proprietary

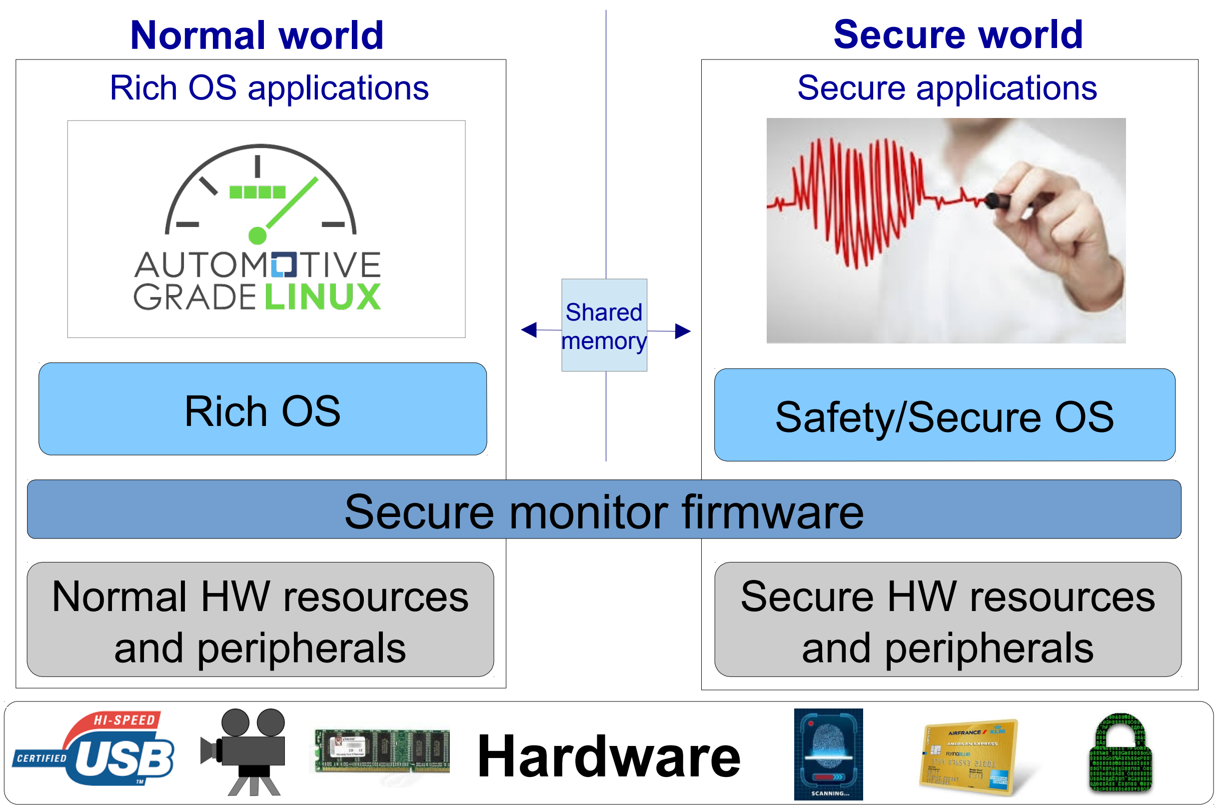
Motivation
Novel software architectures

- Due to the huge growth of processing power the integration of multiple software systems on a single embedded controller becomes a viable option
- The integration of multiple components lowers the number of embedded controllers in the car, consequently…
  - …reduced costs, weight
  - …heat generation and power consumption
- **BUT** these software stacks need to communicate
  - Exchange of information previously over physical BUS in the car
  - Now?



Background
ARMv8-A features: ARM TrustZone

- TrustZone splits processor into two worlds (e.g., Normal world / Secure world)
- Secure monitor firmware (EL3) is needed to support context switching between worlds



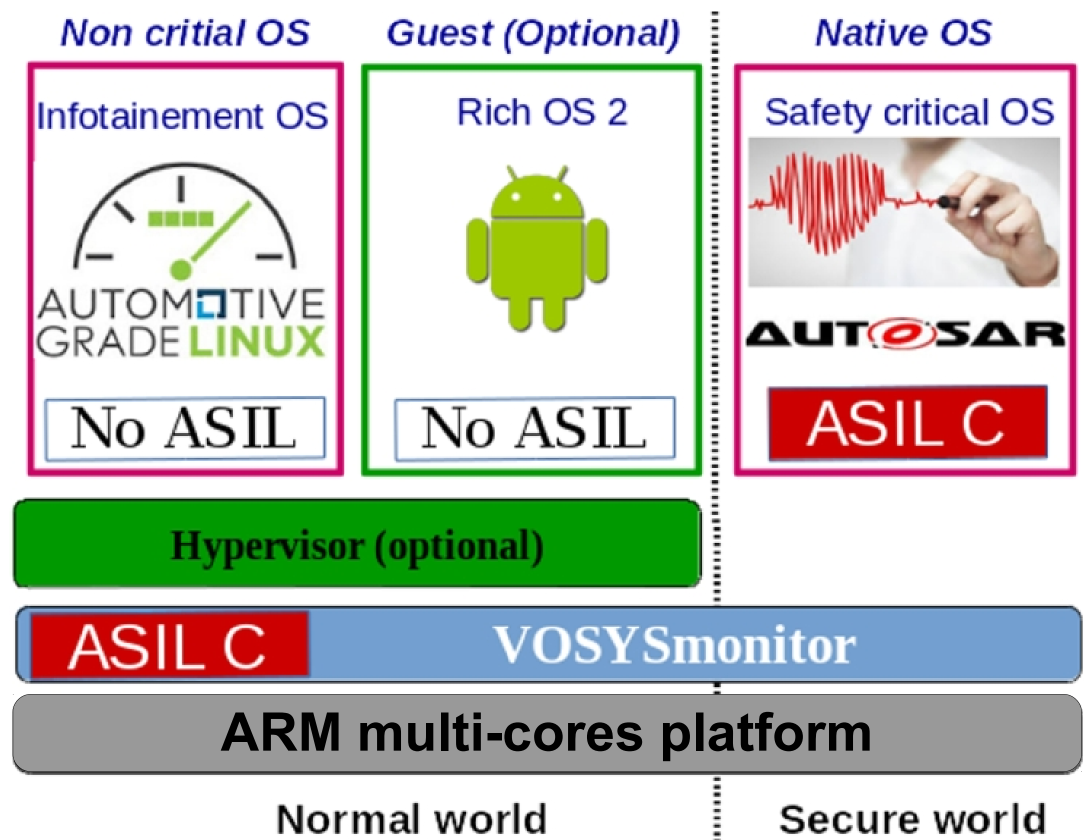
ARM TRUSTZONE
System Security

- Each compartment has access to its own MMU allowing the isolation of Secure and Normal translation tables.
- Caches have tag bits to discern content cached by either secure or normal world.
- Security information is propagated on AXI/AHB bus
- Memory/Peripheral can also be made secure
- Provides secure interrupts

Background
VOSYSmonitor

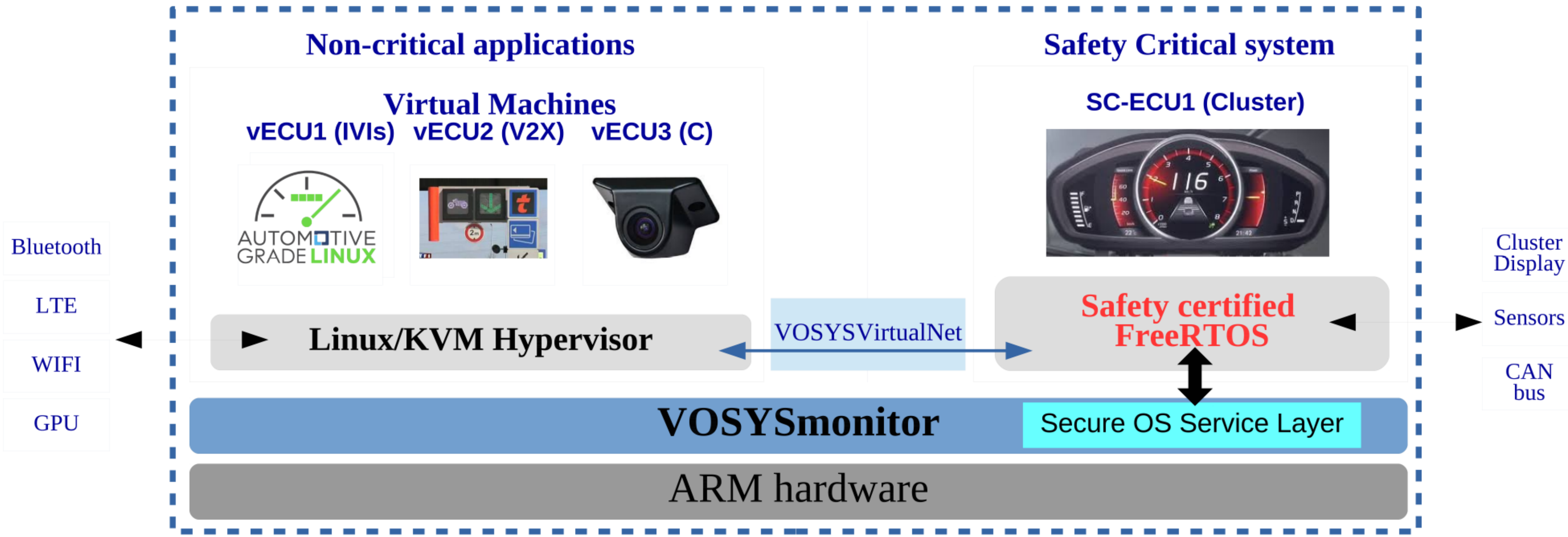
VOSYSmonitor is a Virtual Open Systems proprietary system partitioner (C/ASM), running in the Secure Monitor mode of ARM processors, which enables co-execution of virtualized systems with a safety critical real time OS on the same platform and/or core.

- Certifiable firmware running in secure EL3 mode
- Safety critical RTOS isolation using TrustZone
- Provide virtualization features for non-critical systems
- Power management
- Linux reboot feature
- Modular and scalable architecture



Virtual Open Systems Confidential & Proprietary

VOSYSVirtualNet
Exemplary architecture



Virtual Open Systems Confidential & Proprietary

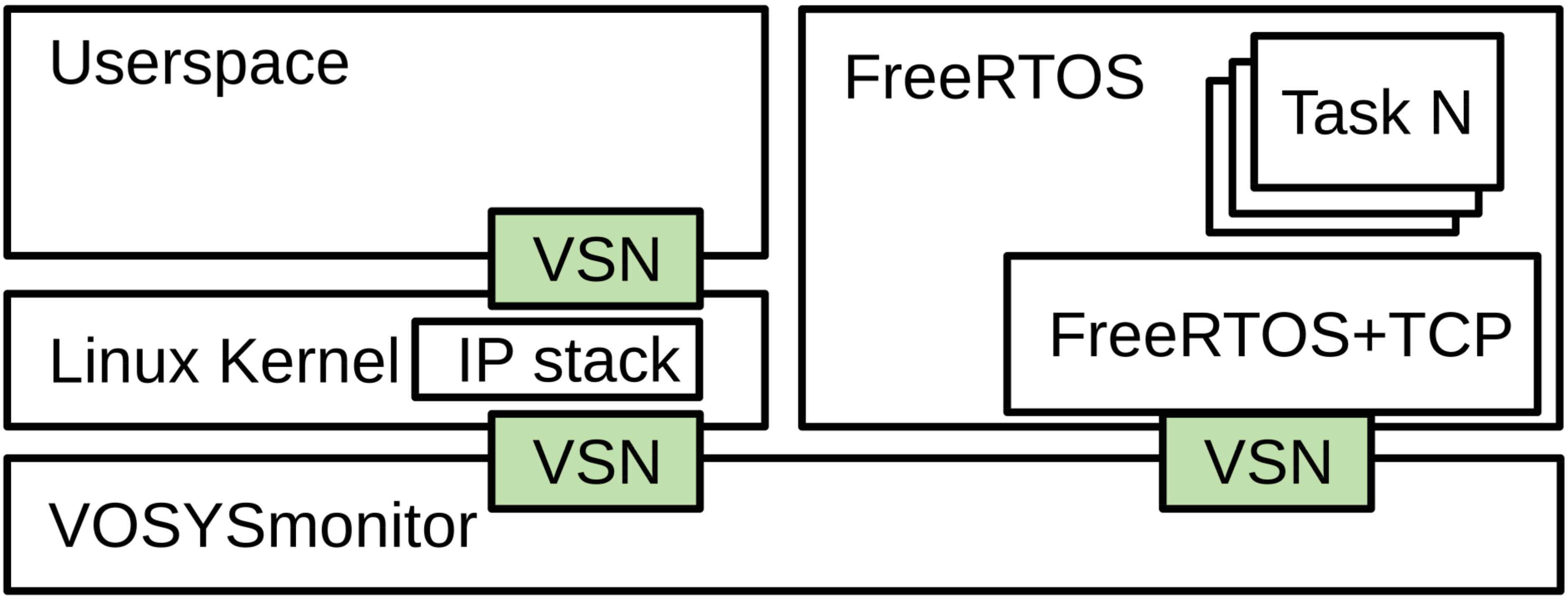
VOSYSVirtualNet Design Goals

Several scenarios raise the need for a communication link between virtual machines (e.g. displaying warning signs, parking assistance information, etc.)

- **Low latency:** Key requirement of our architecture
  - Critical OS, running in the secure world, must forward information to non-secure OS with a low delay
  - All our design decisions were build around this **“low latency”** requirement.
- **Minimally invasive:**
  - Source level access to OSs is not always guaranteed
  - For Linux kernel modifications that can not be made upstream, an external patch has to be maintained constantly.
  - Changes to VOSYSmonitor have to comply to the ISO26262 specification
- **Small hardware requirements:**
  - We need a form of signaling mechanism (e.g. interrupt)
  - But external IRQs are arbitrarily assigned by the SoC
  - So an SGI it is (some of them utilized by Linux kernel, but not all of them)

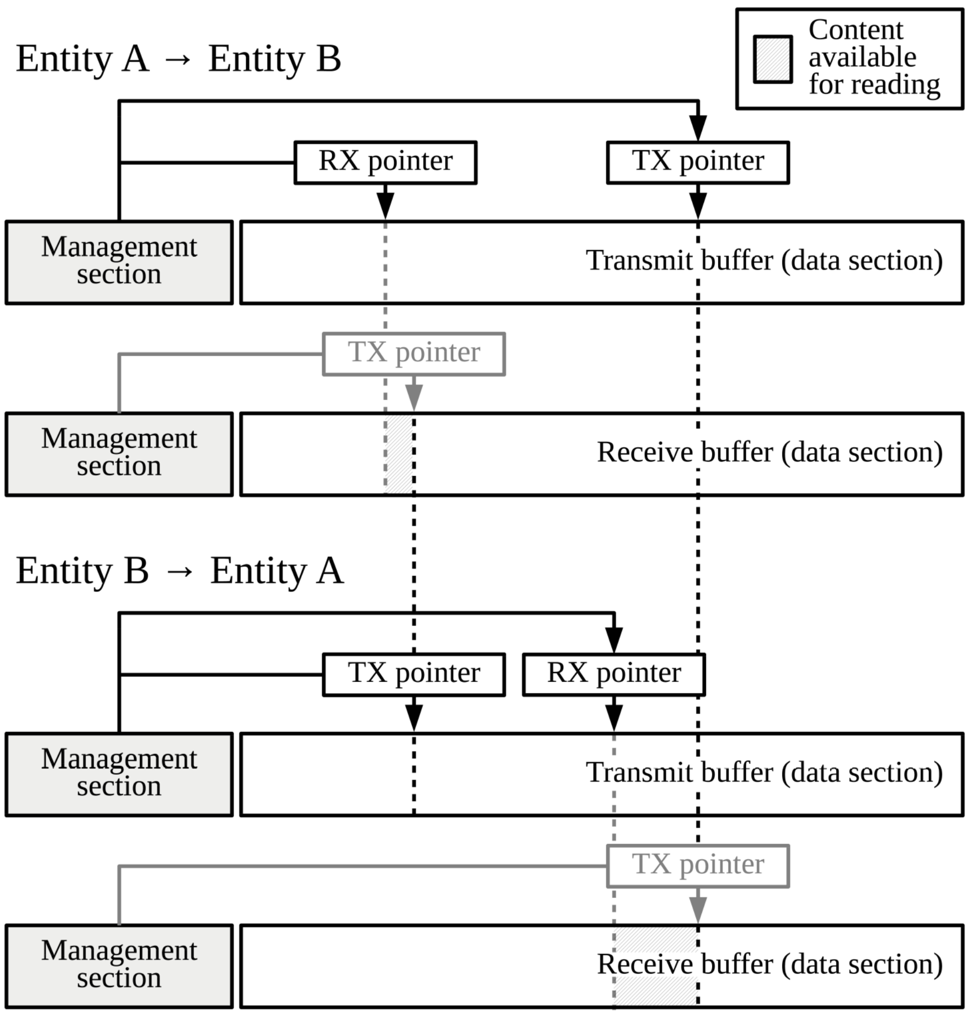
Virtual Open Systems Confidential & Proprietary

VOSYSVirtualNet
Design Goals



Virtual Open Systems Confidential & Proprietary

VOSYSVirtualNet Buffer Layout

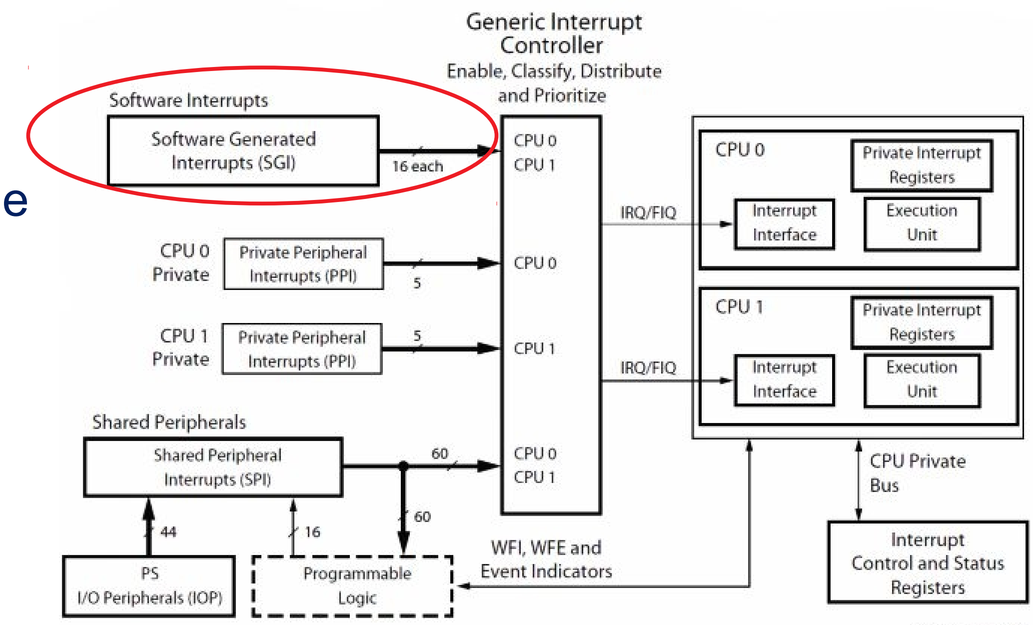


- Simple data structure precedes the transmit (`txbuf`) and receive buffer (`rxbuf`) ("management section")
- Each entity only updates the pointer in its transmit buffer (`txptr`)
- Entity can check if new data is available by comparing `rxbuf.txptr != txbuf.rxptr`
- Then data is copied from the buffer into the respective network stack
  - Linux: `skb_copy_from_linear_data()`
  - FreeRTOS: `memcpy → NetworkBufferDescriptor_t`

Virtual Open Systems Confidential & Proprietary
Virtual Open Systems

VOSYSVirtualNet Signaling

- ARM SoCs usually feature 16 SGIs (Software Generated Interrupt)
- Some of them used by the Linux kernel (e.g. to kick secondary cores)
- Still >8 unused
- When either world sets one of them pending VOSYSmonitor receives a trap
- VOSYSmonitor forwards the request to the other world
- Respective IRQ (SGI) handler is called
  - Linux: `handle_IPI()`
  - FreeRTOS: custom IRQ handler
- Handler schedules lower priority job
  - Linux: `tasklet_schedule()`
  - FreeRTOS: `vTaskNotifyGiveFromISR()`



Virtual Open Systems Confidential & Proprietary

VOSYSVirtualNet Security Measures

- **Denial-of-Service:** Non-critical system sends request at a high rate to critical system to cause a DoS
  - **CM1**: IP stack application of the critical system must be defined with a low priority to allow the execution of other tasks
  - **CM2**: Implement a rate limiter to dismiss requests if they exceed a certain number in a predefined time interval
- **Packet corruption:** Network packets sent by Non-critical system are corrupted by a malicious application
  - **CM1**: IP stack application of the critical system must be isolated in a user application to limit the attack impact
  - **CM2**: Authentication - Cryptographic operations (high overhead)
- **Memory corruption:** Non-critical application allocates buffer at the boundary of Secure world to provoke overwriting of critical system
  - **CM1**: Non-critical system cannot directly access the buffer area located in the Secure world
  - **CM2**: Sanity check can be performed on the buffer location to ensure that the Secure world is not affected

Virtual Open Systems Confidential & Proprietary

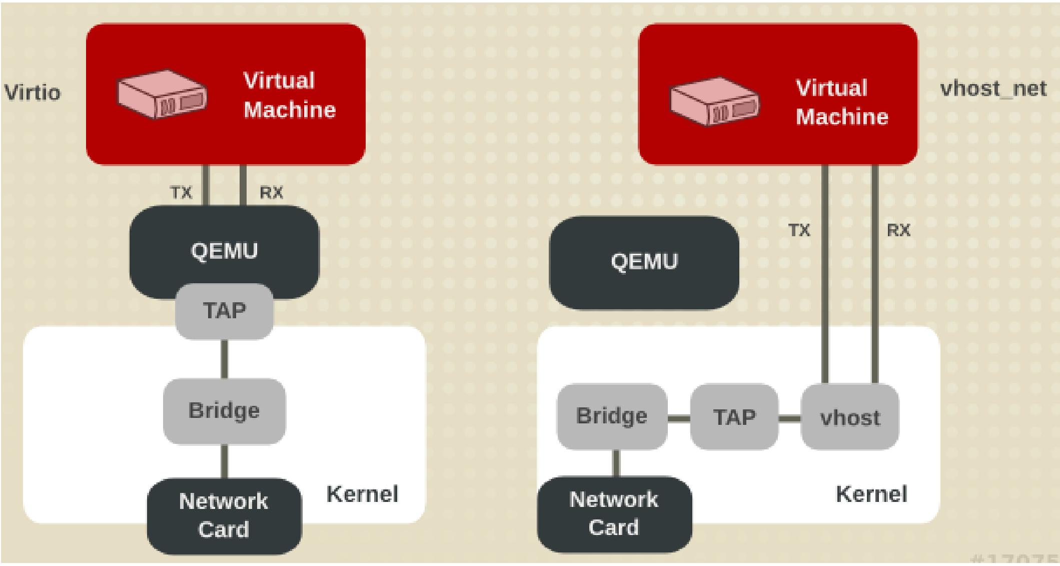
Results
Code Statistics

- FreeRTOS
  - Device driver: 267 (SLOC/ANSCI C)
  - Modifications to FreeRTOS: -
  - Modifications to FreeRTOS+TCP: -
- Linux
  - Device driver: 321 (SLOC/ANSCI C)
  - Modifications to Linux kernel: 8 (SLOC/ANSI C)

Virtual Open Systems Confidential & Proprietary

Results
Measurement Setups

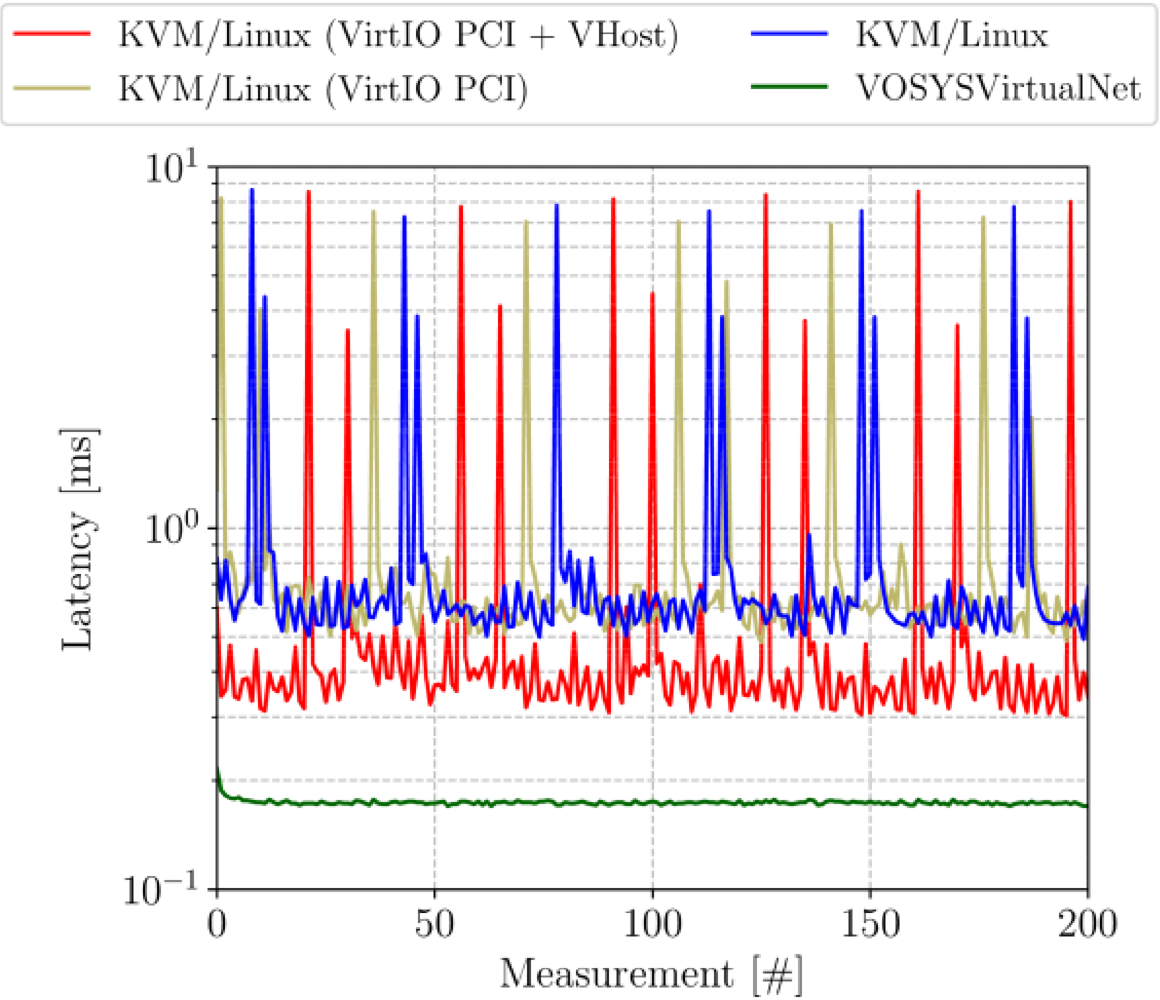
- VOSYSVirtualNet
  - Linux ↔ FreeRTOS shared memory network link
  - Native driver that puts/gets the packets on/from the *"wire"* (buffer)
  - Functions `skb_copy_from_linear_data()` and `skb_copy_to_linear_data()` involve `memcpy()`
- Linux KVM / VirtIO (+ PCI + VHost)
  - Linux (Qemu) ↔ Linux (Qemu) queue-based network link
  - Para-virtualized driver that writes into queues (VirtQueue)
  - Queues are part of guest memory
  - Qemu (or host kernel in case of Vhost) can directly write into queues (zero-copy)



- https://access.redhat.com: *"vhost_net moves part of the Virtio driver from the user space into the kernel. This reduces copy operations, lowers latency and CPU usage."*

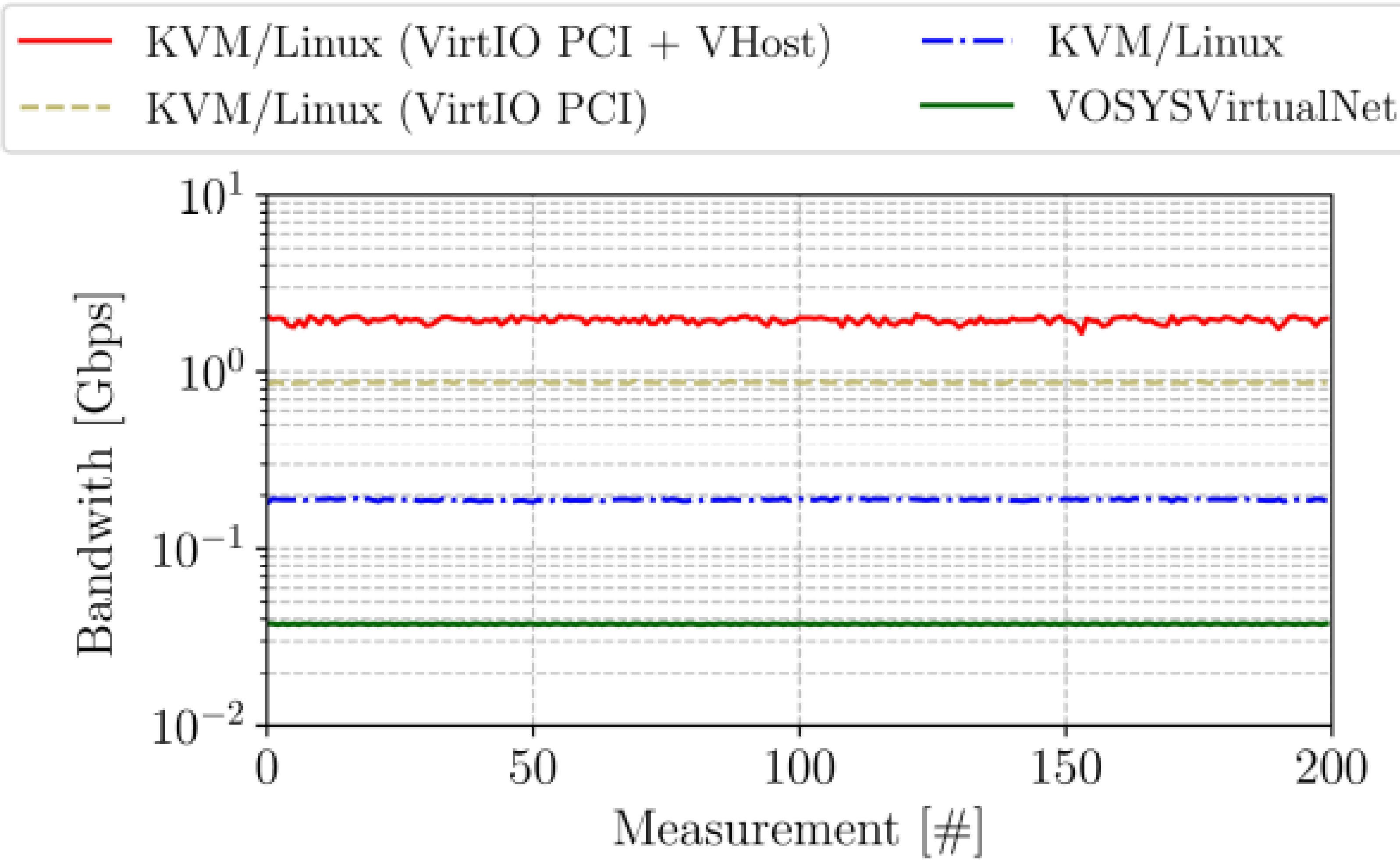
Virtual Open Systems Confidential & Proprietary

Results
Ping latency measurements

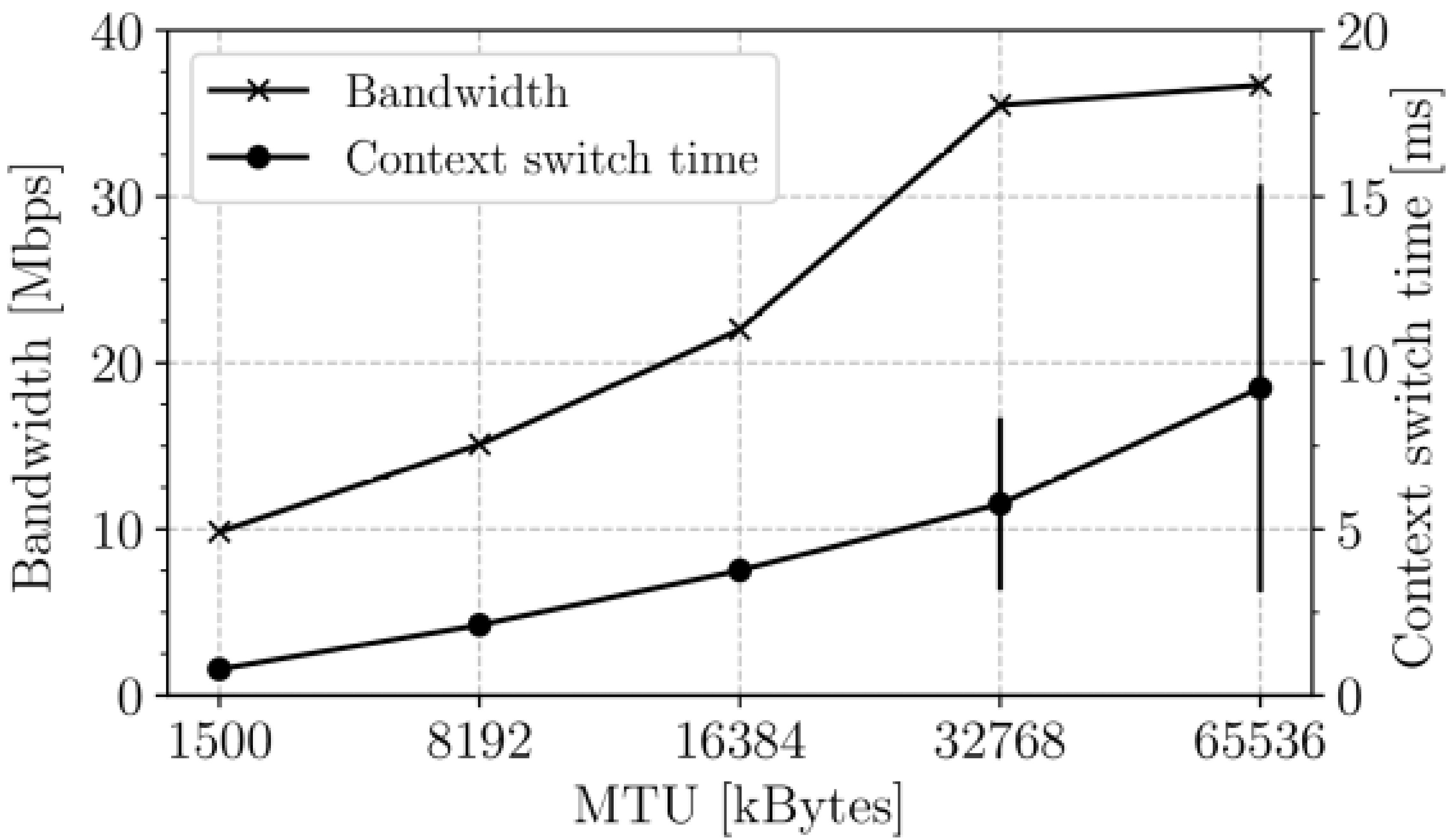


Virtual Open Systems Confidential & Proprietary

Results
IPerf TCP bandwidth measurements



Results
MTU vs. achievable bandwidth



Virtual Open Systems Confidential & Proprietary

Conclusion
Thank you!

- **Future work**
  - For the demonstrator, we choose FreeRTOS running in the secure world, but integration of our design into a fully AUTOSAR compliant OS still stands out
  - Low bandwidth results need investigation
    - Achieved a bandwidth of ~38Mbps (but, implementation in the secure world is very OS specific)
    - Tests with increased MTU sizes already suggest that higher bandwidth results can be achieved in the current setting
    - Utilize Large Receive Offload (LRO)

- **Conclusion**
  - Approach feasible (full prototype on an ARM Juno development board)
  - Achieved good results in terms of latency

Virtual Open Systems Confidential & Proprietary

contact@virtualopensystems.com
Web: virtualopensystems.com
Products: http://www.virtualopensystems.com/en/products/
Demos: virtualopensystems.com/en/solutions/demos/
Guides: virtualopensystems.com/en/solutions/guides/
Research projects: virtualopensystems.com/en/research/innovation-projects/