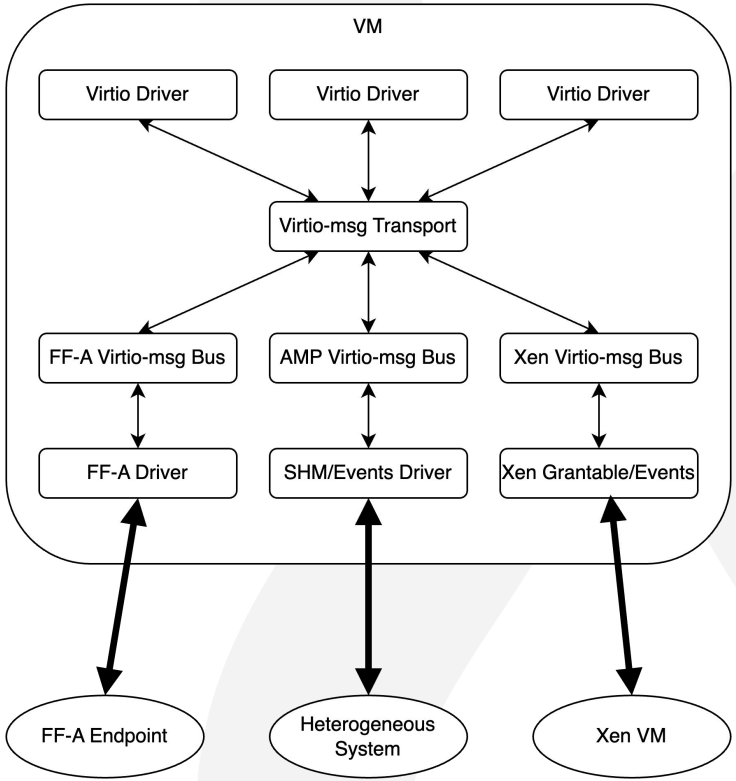
linaro Connect 2025

Virtio Meets ARM FF-A: Bridging Normal and Secure Worlds in Virtualized Systems

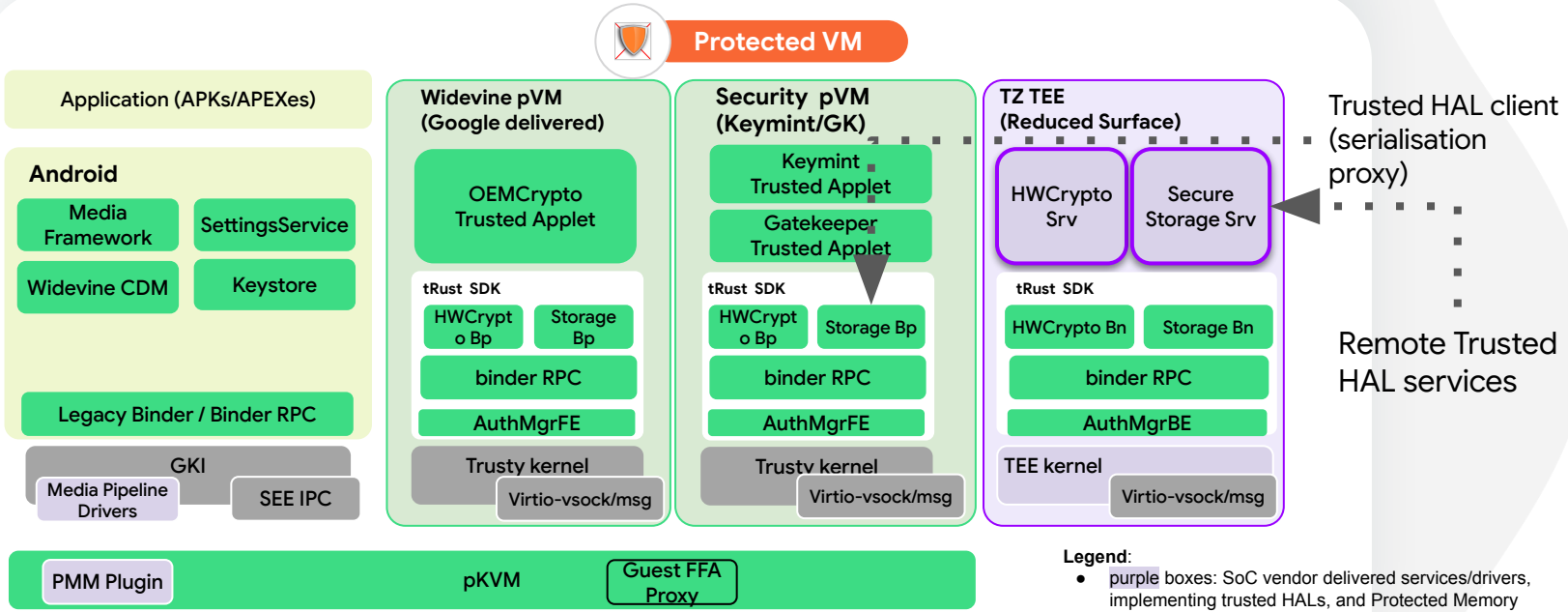
Bertrand Marquis / Viresh Kumar
ARM / Linaro

Virtio-msg

- New Virtio Transport
  - along side PCI, MMIO and Channel I/O
  - Message based
  - Large range of Targets
    - Heterogeneous, Secure/Non-secure, VM to VM, …etc
  - More details in Bill Mills presentation
    - “Virtio-msg: Making virtio work anywhere”
- Layer based design
  - Virtio-msg Transport
    - Request to Message
    - Agnostic of the communication medium
  - Virtio-msg Bus
    - Communication specific
      - IPC, Hypercall, mailbox, Xen, FF-A, …etc
    - Roles:
      - Bus and Device discovery
      - Bus and Device lifecycle (hotplug)
      - Optional DMA management
      - Message transport between Frontend and Backend



Virtio-msg: Android 16 use case

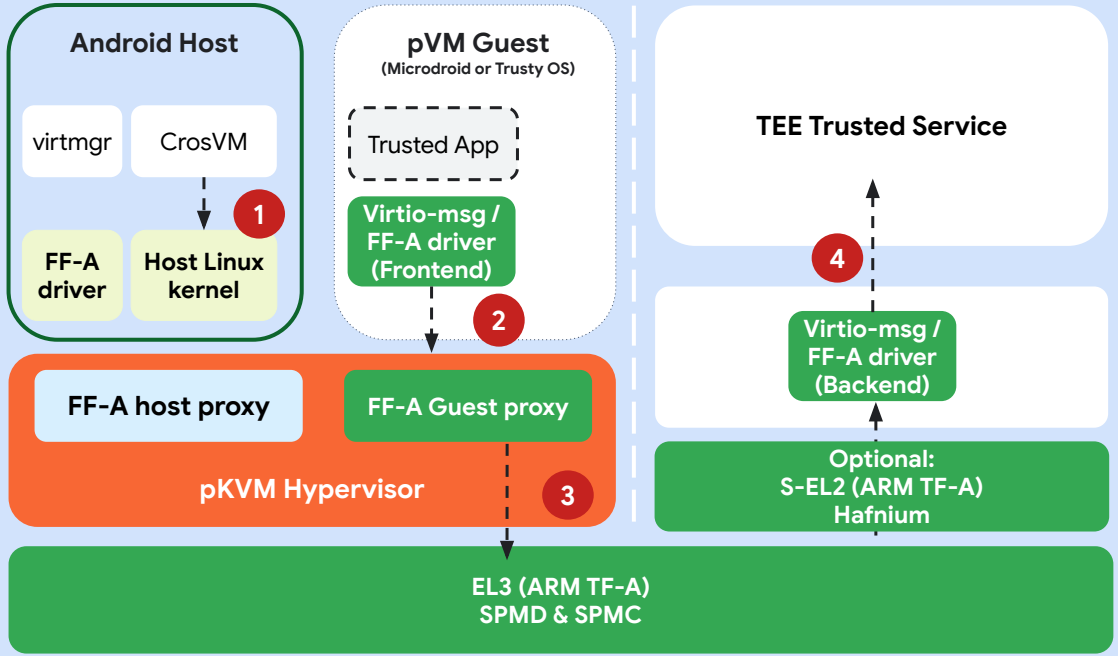


Legend:

- purple boxes: SoC vendor delivered services/drivers, implementing trusted HALs, and Protected Memory Manager
- orange boxes: OEM delivered TA (Innovation)
- green boxes: Android delivered (Services, Applets, tRust SDK)
- blue boxes: Standardised linux or Trusty driver (upstream)

- Bp: stands for Binder Proxy; i.e client side of the IPC
- Bn: stands for Binder Native; i.e service side of the IPC

Virtio-msg: VM to TZ in Android 16



What's going on ?

1. Android Host starts a VM
2. The guest VM probes the FF-A driver
3. The driver establishes a shared virtqueue with TEE (Trusty OS)
4. A new IPC link is established over the shared mem virtqueue with TEE. The app opens **/dev/tipc** and apps can talk across domains !

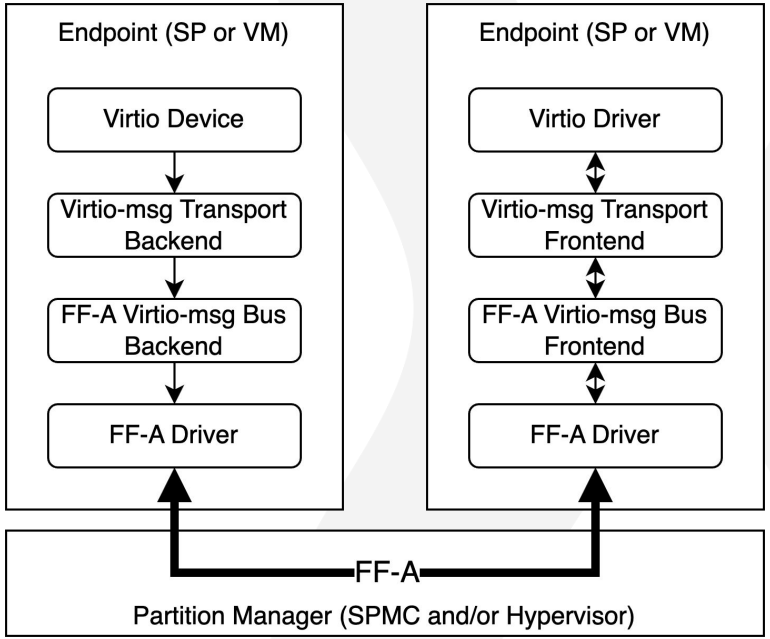
QEMU testbench shared on aosp

Virtio-msg: FF-A 1.2 features used

- Protocol UUID-based messaging/identification
  - FF-A endpoints identified using a protocol UUID
  - UUID based addressing for communication
- FF-A direct or indirect messaging
  - Transmit protocol messages
  - Bus discovery, events and hotplug
- Memory sharing
  - FF-A memory sharing operations used map driver memory in device
  - Virtio Virtqueues and DMA use FF-A memory sharing

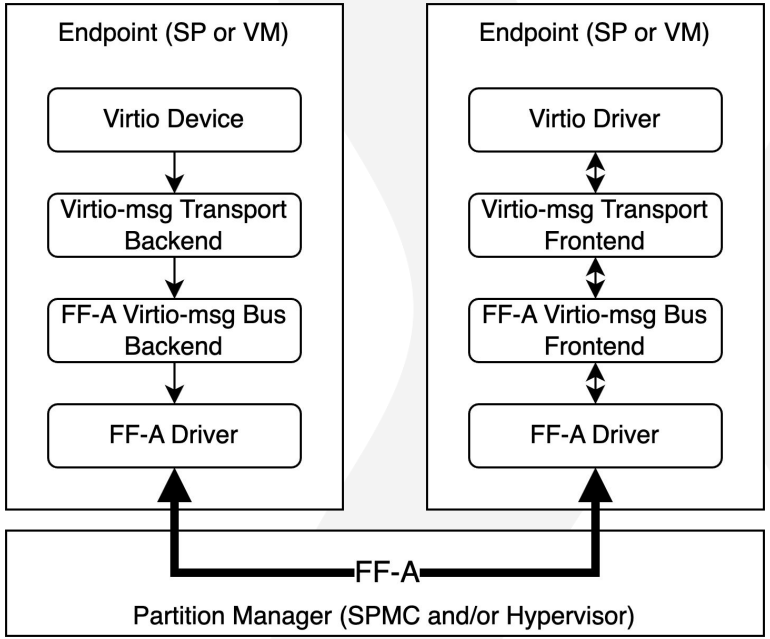
Virtio-msg: FF-A as Bus

- Discovery by FF-A Virtio-msg Frontend
  - Finds endpoint with the FF-A Virtio-msg backend UUID
    - FF-A ABI: FFA_PARTITION_INFO_GET
  - Device Messaging type to use
    - Direct or Indirect
    - From partition info
  - Retrieve FF-A backend information
    - Virtio-msg: FFA_BUS_MSG_VERSION
  - List devices in endpoint
    - Virtio-msg: BUS_MSG_GET_DEVICES
  - Get device information
    - Virtio-msg: VIRTIO_MSG_GET_DEVICE_INFO
  - Register device
    - Probe from Virtio Driver
- Relay messages from Transport
  - Using direct or indirect messages
  - Driver config access, virtqueue configuration
  - Events from driver/device



Virtio-msg: FF-A as Bus

- Memory sharing from Frontend
  - DMA map requests from Driver/Transport
  - Share memory with backend endpoint
    - FF-A ABI: FFA_MEM_SHARE
  - Signal memory shared to backend
    - Virtio-msg: FFA_BUS_MSG_AREA_SHARE
  - Backend maps memory
    - FF-A ABI: FFA_MEM_RETRIEVE
  - Unshare once done
    - Virtio-msg: FFA_BUS_MSG_AREA_UNSHARE
    - FF-A ABI: FFA_MEM_RELINQUISH
    - FF-A ABI: FFA_MEM_RECLAIM
- Device hotplug/unplug
  - Signal to frontend devices added/removed
    - Virtio-msg: BUS_MSG_DEVICE_ADDED
    - Virtio-msg: BUS_MSG_DEVICE_REMOVED



Status Update

Early Development Challenges

linaro Connect

- No infrastructure available to test virtio-msg transport with FF-A.
- Reused prior setup from Project Orko / Stratos
  - Hypervisor agnostic Rust based Virtio backends (github: rust-vmm)
  - Xen setup with Xen-vhost-frontend (Rust, MMIO support)
- Developed a generic virtio-msg transport layer in Linux
  - Implemented MMIO bus layer
  - MMIO traps in the configuration space to communicate with backends
- Unblocked subsequent development providing a generic virtio-msg implementation

Virtio-msg over FF-A (Xen)

- FF-A Indirect message support in Xen (Bertrand Marquis)
- Developed the Virtio-msg FF-A bus implementation in Linux
- Required FF-A Indirect messages support in Linux
  - Out of tree patches (Sudeep Holla / ARM)
  - Numerous fixes over that
- Tested Rust based I2C / GPIO backends (github: rust-vmm)
- Memory sharing - pending

Virtio-msg over FF-A (Android / Trusty)

linaro Connect

- Android Virtualization Framework (AVF)
  - Running isolated virtual machines
  - pVM (pKVM) / Microdroid guests
- Secure Partition Manager (SPM) for running secure partitions, two options:
  - Hafnium in S-EL2
  - EL3 SPMC for devices that don’t support S-EL2
- Trusty - Secure partition (SP)
- Virtio-msg (FF-A) to communicate between pVM / SP and Host / SP
- Tested Virtio-msg with FF-A bus (Direct messages) between Host / SP and pVM / SP
- Tested FF-A based memory sharing
- Google verified:
  - pVM (Trusty) / SP (Trusty) - non Linux
  - WIP: pVM (Microdroid) / SP (Trusty)

Virtio-msg: FF-A memory sharing

linaro Connect

- FF-A based memory sharing between endpoints.
- Dynamic mapping
  - Advance mapping for virtqueues (dma_alloc_coherent)
  - On demand for buffer (kmalloc) mapping
  - High runtime overhead for small buffers

- Static mapping
  - “reserved-memory” in DT (“restricted-dma-pool”)
  - “memory-region” in FF-A device node
  - Large enough chunk of memory mapped at initialization
  - DMA coherent allocations from this region
  - On demand buffers (kmalloc) are bounced (memcpy) from this region
  - Potentially unused memory
  - Low runtime cost

Thank You!