SIEMENS
Ingenuity for life

Reworking the Inter-VM Shared Memory Device

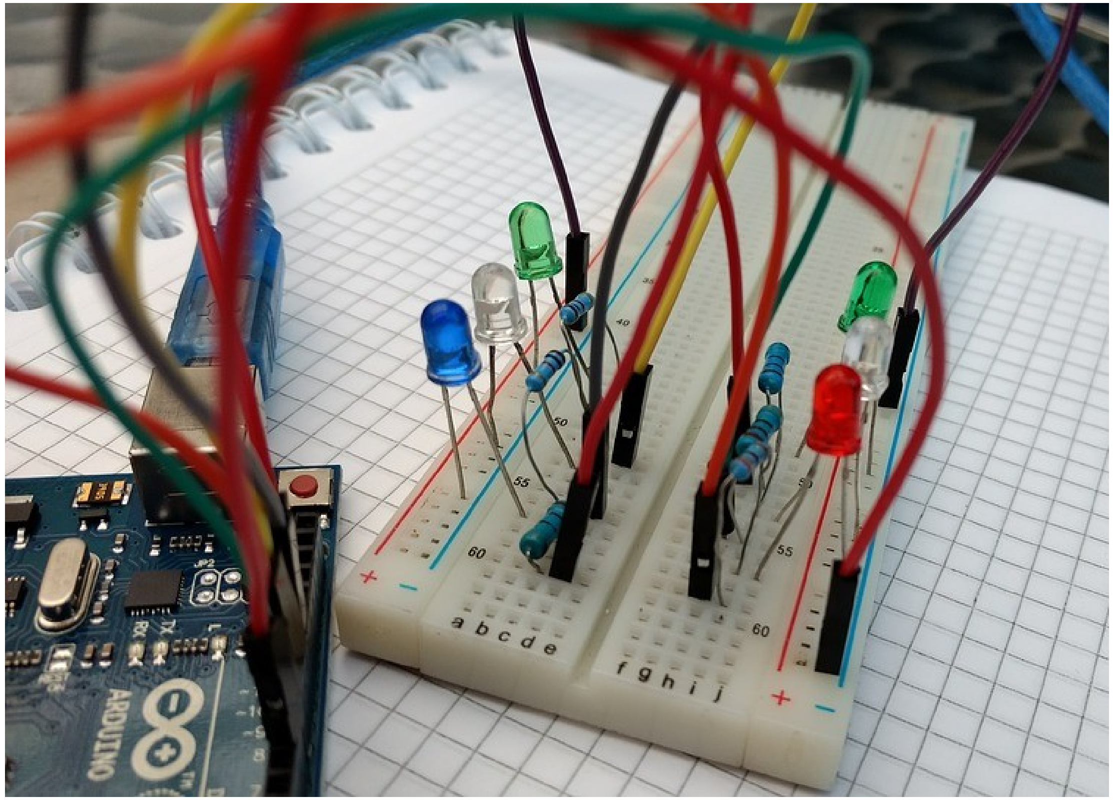
Jan Kiszka | KVM Forum 2019

Agenda

- Use cases for shared memory devices
- Nahanni / IVSHMEM 1.0
- Deficits of current approach
- IVSHMEM 2.0
- First implementations and drivers
- (Re-)using Virtio with shared memory
- How to move forward?

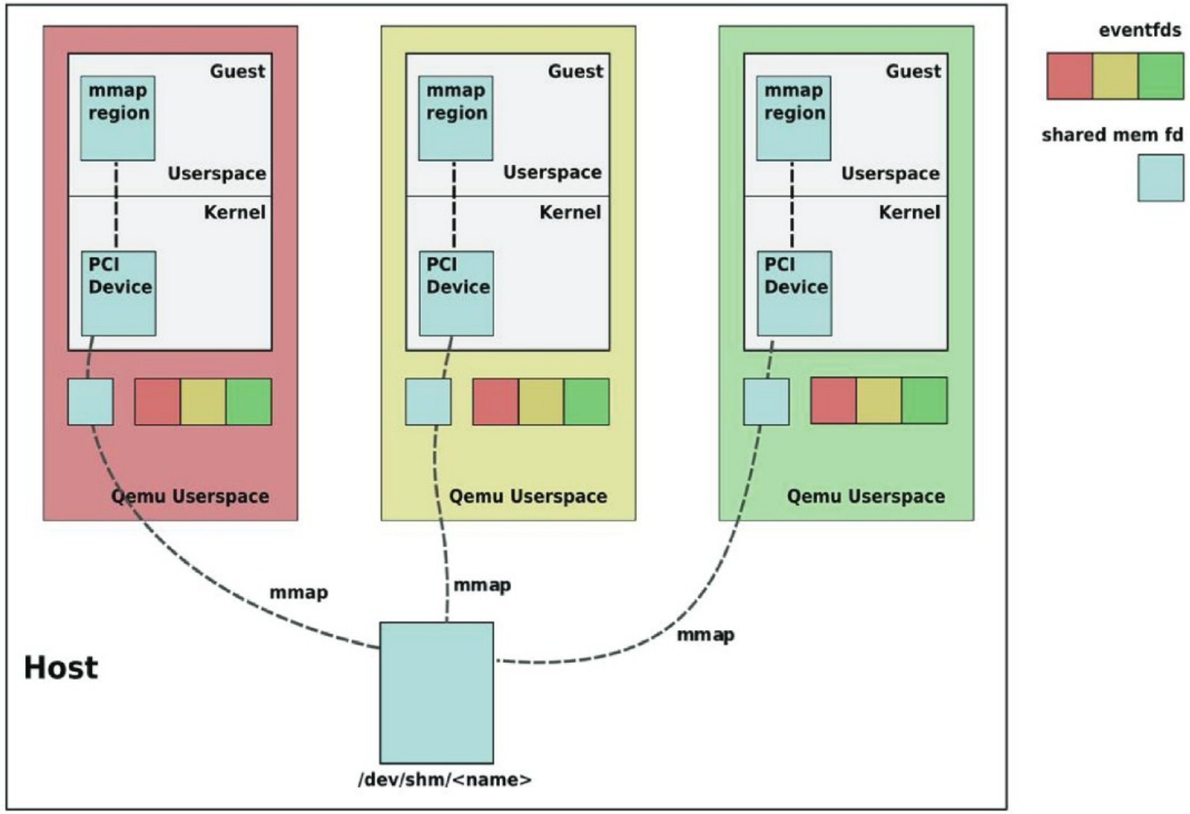
Shared Memory Devices – The Breadboard Boards of Virtual Devices

- Easy to define (still hard to get right)
- Simple security model
  → easy to export to guest applications
- Simple drivers
  → helps with non-Linux guests
- Allows to map custom / legacy protocols to the virtual world
- Logical step from Inter-Process to Inter-VM Communication
- Used by many embedded hypervisor
- ...and even for “The Machine”
  (Fabric-Attached Memory Emulation in QEMU)



From Nahanni to QEMU IVSHMEM

- Designed by Cam Mcdonell & colleagues for fast cross-VM IPC and process-level data sharing
- Merged as ivshmem into QEMU 0.14
- PCI device
- Two modes
  - Just shared memory
  - SHMEM + signaling
- Second mode requires server process



Deficits of IVSHMEM Design and Implementation

- No life-cycle management
  - Generic state exchange between peers
  - Notification about disappearing peers
- Peer doorbell (signaling) support only optional
- Interrupt handling not optimized for virtual device scenario
- No support for uni-directional shared memory (r/w by one peer, r/o by others)
- Missing upstream driver support
- QEMU server implementation says:
  ”Example code, do not use in production”

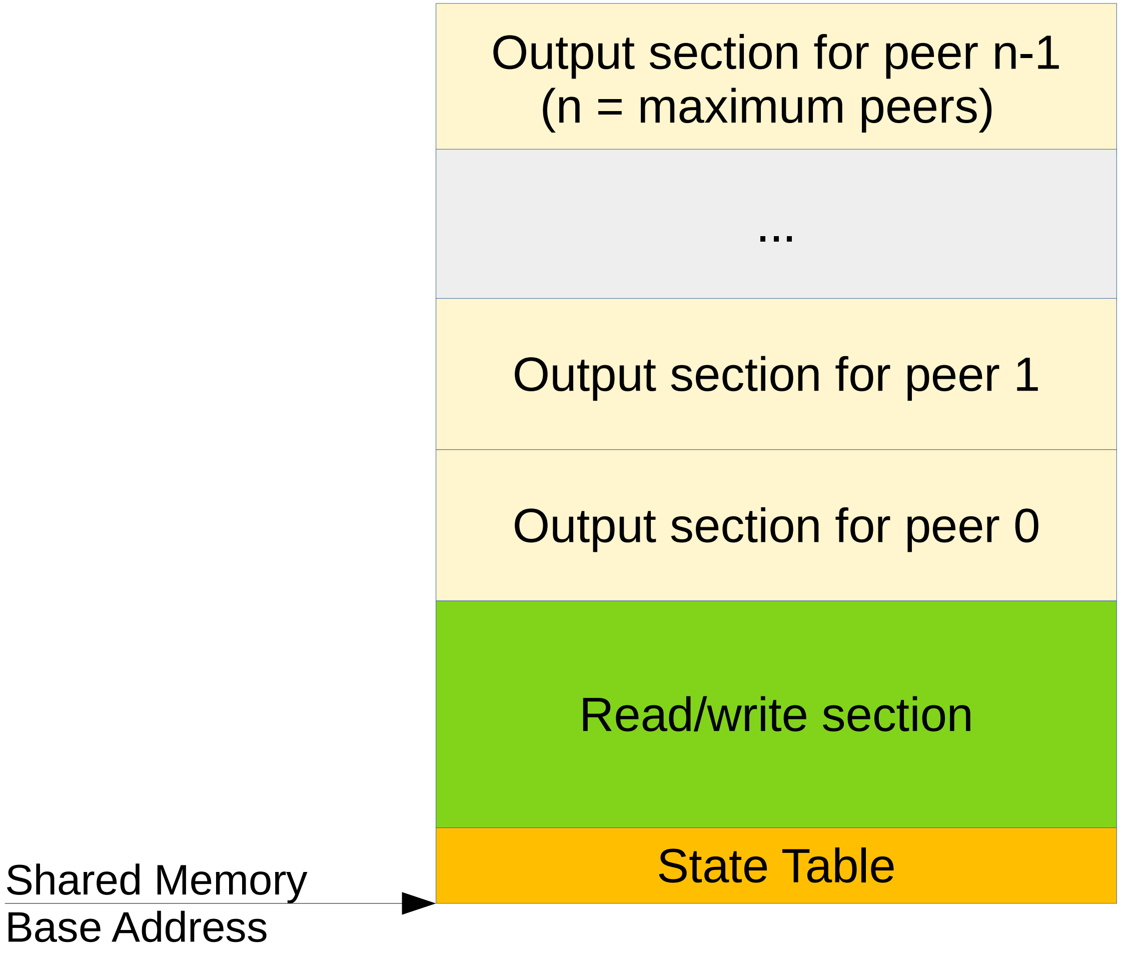
IVSHMEM 2.0 – Key Differences

- Hypervisor-backed peer life-cycle tracking
- Interrupt support with same maximum number of vectors is mandatory
- Only edge-triggered interrupts without any status register
- Efficient unprivileged access possible (e.g. via UIO)
- Protocol ID propagation to peers using PCI class and interface (enables driver probing)
- Uni-directional shared memory sections (optional)
- Static shared memory location (optional)

New!

Shared Memory Sections

- Table of peer states
  - Read-only for guests
  - Updated by Hypervisor on state register writes and guest reset / disappearance
  - State changes also trigger interrupts (vector 0)
- Read/write section
  - Shared by all peers
  - Size can be zero
- Output sections
  - All have the same size (can be zero)
  - Only writable for owning guest
  - Read-only for others



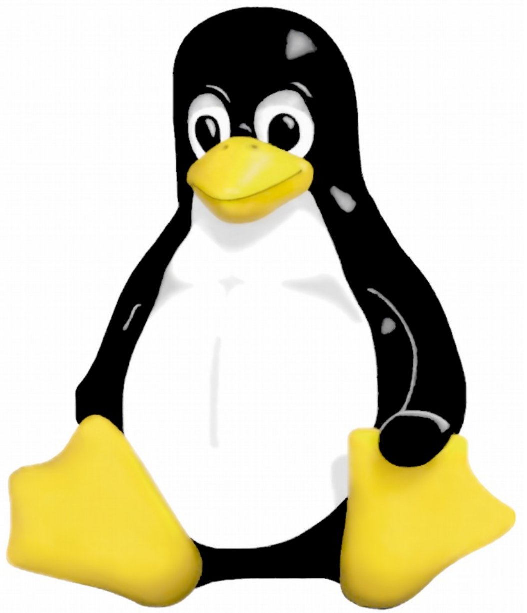
Output section for peer n-1
(n = maximum peers)
...
Output section for peer 1
Output section for peer 0
Read/write section
State Table
Shared Memory
Base Address

IVSHMEM 2.0 Implementations

- QEMU
  - New device: ivshmem2
  - New server: ivshmem2-server
  - Server now also defines
    - Output section size
    - Maximum number of peers
    - Number of interrupt vectors
    - Protocol ID
- Jailhouse
  - Only statically located shared memory
  - <500 LoC (x86, ARM, ARM64)
  - Shall remain the only virtual device

Linux Drivers for IVSHMEM 2.0

- UIO
  - Complete rewrite of original uio_ivshmem
  - Supports proper interrupt throttling
  - Listens on all possible interrupt vectors (but coalesces them – UIO interface limitation)
  - Exports all shared memory sections separately, respecting read-only properties
  - Zeroes state on userspace disconnect
- ivshmem-net
  - Yet another peer-to-peer virtual Ethernet
  - Developed for Jailhouse's IVSHMEM variant
  - Uses virtio rings internally
  - Mapped on IVSHMEM 2.0 for demonstration and testing purposes (incl. uni-directional shmem)
  - Will likely be superseded by virtio-net



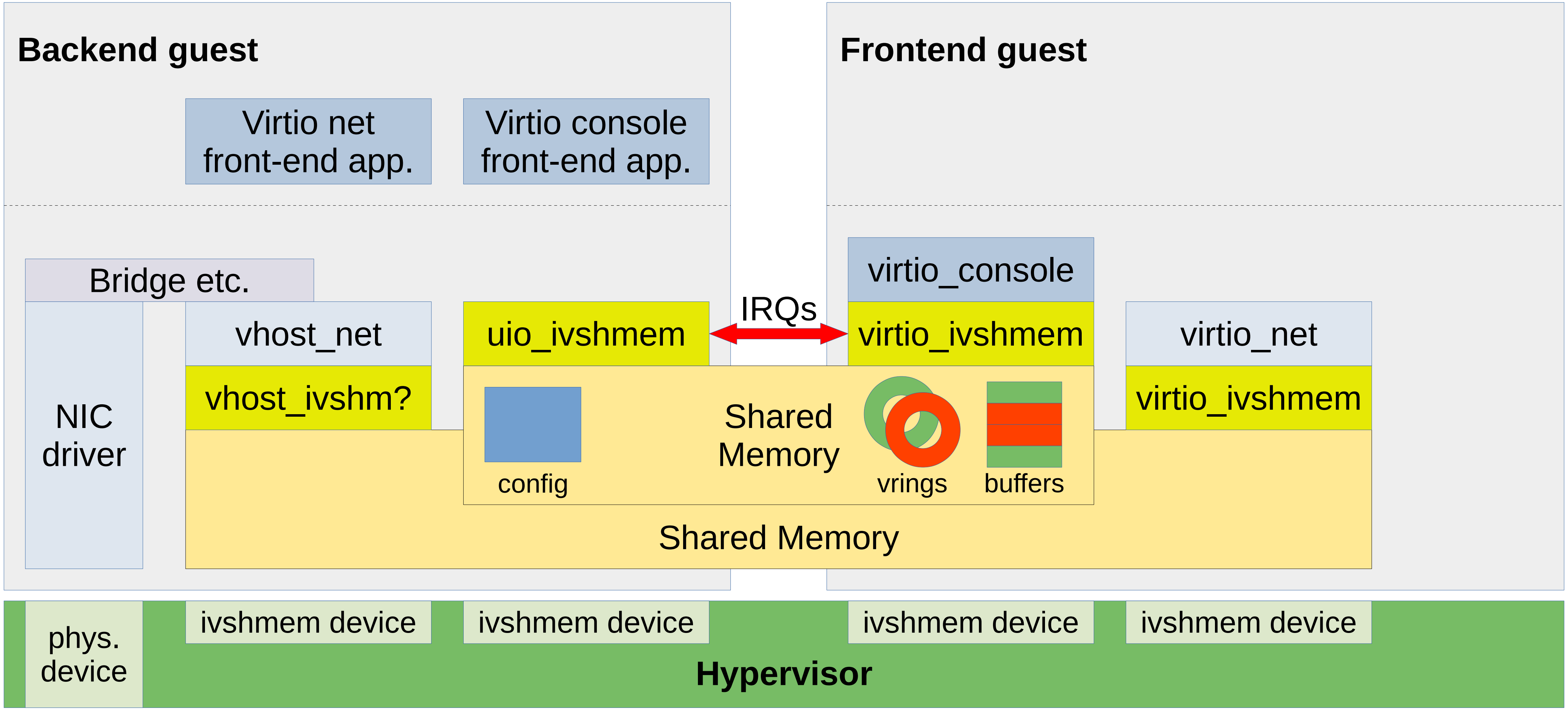
Using VIRTIO with IVSHMEM

- Why do we want this?
  - Scenario: IVSHMEM as only device
  - Protocols needed for higher-level devices (network, storage, serial/console, …)
  - Do not reinvent, better reuse existing protocols
  - ...and drivers
- How to implement it?
  - Define new VIRTIO transport “shared memory”, generically or concretely over IVSHMEM 2.0
  - Map all data (vrings, buffers) into per device shared memory (GPA → shared memory offset)
  - Map configuration and “registers” into same shared memory
  - Proof-of-concept implementation for Linux exists, specification still to-do

OASIS

Virtual I/O Device

VIRTIO over IVSHMEM Overview



Configuration Space via virtio-ivshmem

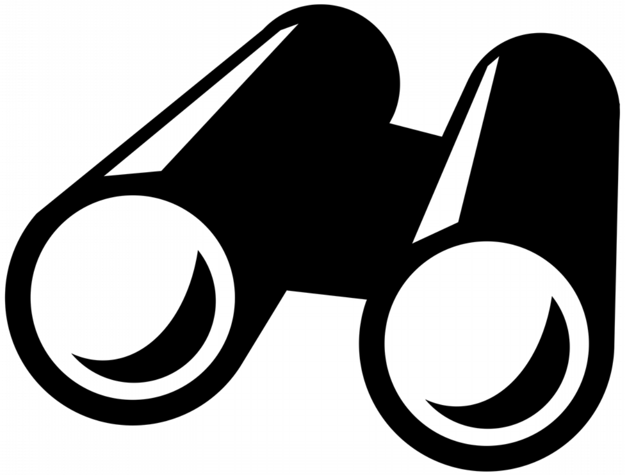
- Data structure at the beginning of r/w section
- “Register”/config write emulation
  - Drivers waits until write_transaction is 0
  - Driver writes new register value into shared memory
  - Driver writes register offset into write_transaction
  - Driver send vector 0 to device peer
  - Device processes written value and zeroes write_transaction
- Adjusted semantic of config_generation
  - Odd value: update in progress
  - Even value: update completed

```
struct virtio_ivshmem_header {
    __le32 revision;
    __le32 size;
    __le32 write_transaction;
    __le32 device_features;
    __le32 device_features_sel;
    __le32 driver_features;
    __le32 driver_features_sel;
    __le32 queue_sel;
    __le16 queue_size;
    __le16 queue_device_vector;
    __le16 queue_driver_vector;
    __le16 queue_enable;
    __le64 queue_desc;
    __le64 queue_driver;
    __le64 queue_device;
    __u8 config_event;
    __u8 queue_event;
    __u8 __reserved[2];
    __le32 device_status;
    __le32 config_generation;
    __u8 config[];
};
```

How to Move Forward?

SIEMENS
Ingenuity for life

- Our primary goals
  - Establish a standard for shared memory devices
  - Make shared memory official virtio transport
- Options
  - virtio-over-abstract-shmem
  - virtio-over-ivshmem
  - ivshmem 2.0 spec maintained inside QEMU (and used by Jailhouse)
  - ivshmem 2.0 spec remains Jailhouse-specific (likely under different name)
  - ivshmem 2.0 becomes virtio device
- Our offerings
  - Enhance specification based on feedback
  - Contribute and maintain ivshmem 2.0 in QEMU (including server)



Resources

- IVSHMEMv2 specification:
  https://github.com/siemens/jailhouse/blob/wip/ivshmem2/Documentation/ivshmem-v2-specification.md
- IVSHMEMv2 device for QEMU:
  http://git.kiszka.org/?p=qemu.git;a=commitdiff;h=wip/ivshmem2
- IVSHMEMv2 for Jailhouse:
  https://github.com/siemens/jailhouse/commits/wip/ivshmem2
- Linux driver uio_ivshmem:
  http://git.kiszka.org/?p=linux.git;a=blob;f=drivers/uio/uio_ivshmem.c;hb=queues/jailhouse-ivshmem2
- Linux driver ivshmem-net:
  http://git.kiszka.org/?p=linux.git;a=blob;f=drivers/net/ivshmem-net.c;hb=queues/jailhouse-ivshmem2
- Linux driver ivshmem_virtio:
  http://git.kiszka.org/?p=linux.git;a=blob;f=drivers/virtio/virtio_ivshmem.c;hb=queues/jailhouse-ivshmem2
- virtio-console back-end via uio_ivshmem:
  http://git.kiszka.org/?p=linux.git;a=blob;f=tools/virtio/virtio-ivshmem-console.c;hb=queues/jailhouse-ivshmem2

Thank You! Any Questions?

SIEMENS
Ingenuity for life

Jan Kiszka
Siemens Corporate Technology

E-mail:
jan.kiszka@siemens.com

siemens.com