Porting virtio to PowerVM Hypervisors

KVM Forum 2010

Ricardo Marin Matinata – rmm@br.ibm.com

LTC Brazil
IBM Linux Technology Center

August 2010

Expectations

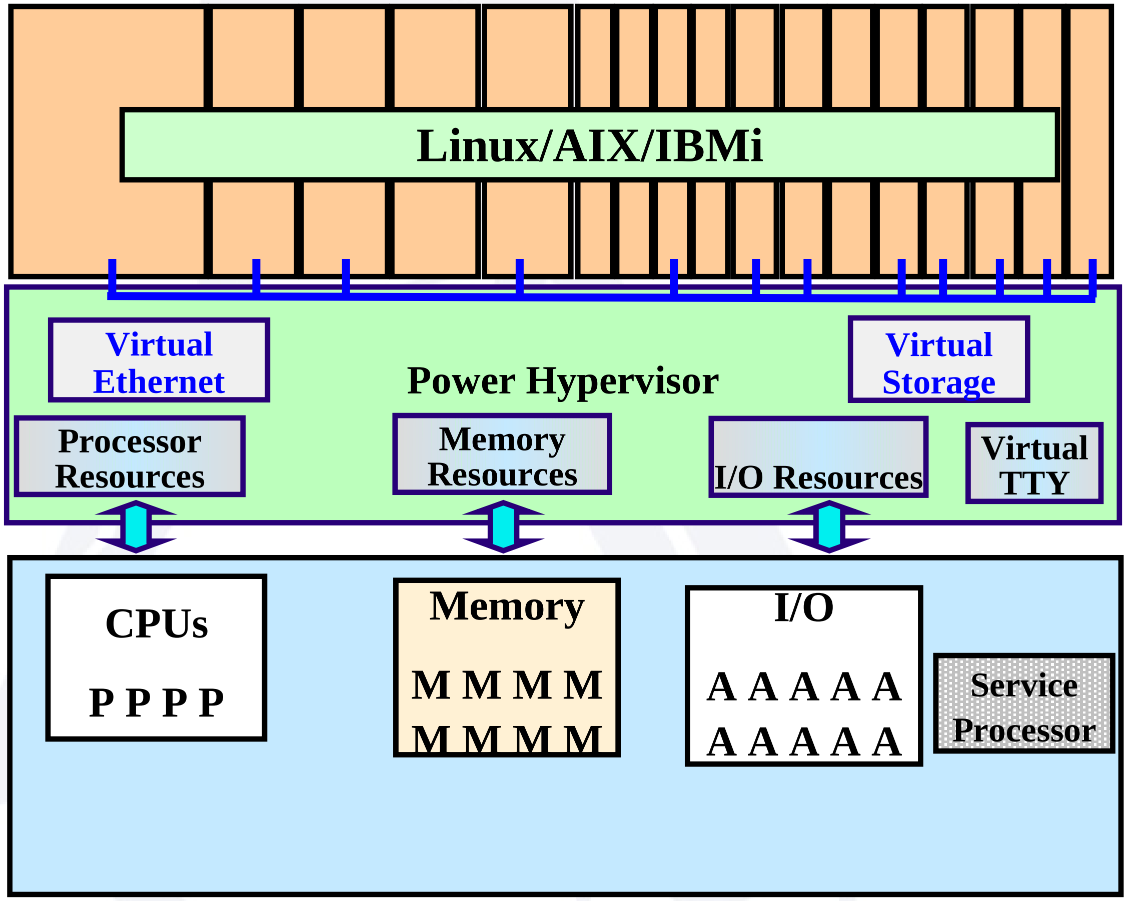
- Background
- Motivations
- Virtual I/O on POWERVM
- Implementation Strategy
- Device Configuration
- Virtqueues

Background

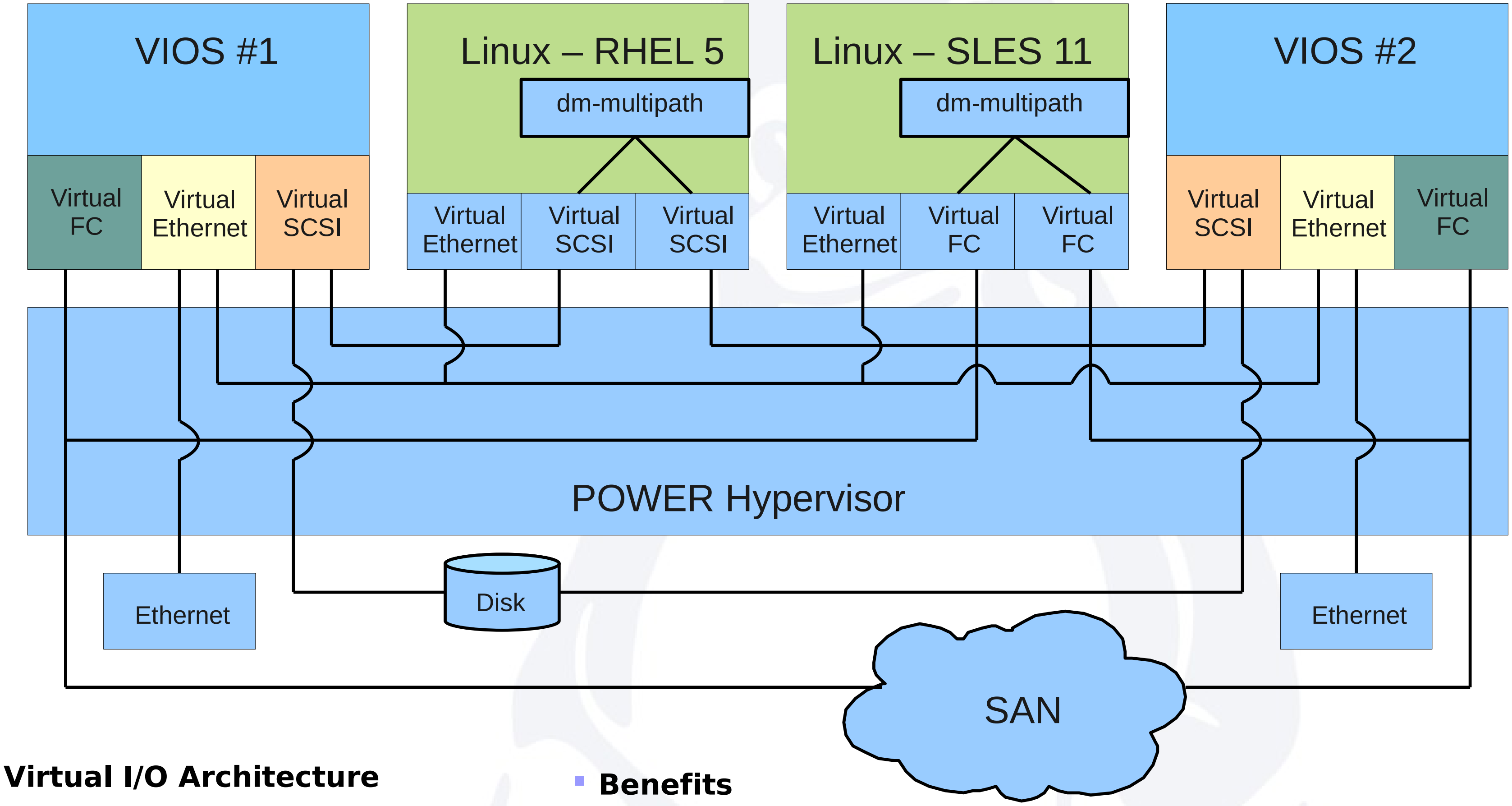
- IBM Brazil entitled to incentive grants in Brazil, related to manufacturing of POWER Systems locally
  - Has to be POWER Systems related
  - Strong research “appeal”
- Execution under responsibility of IBM LTC Brazil (arquitecture & PM), in partnership with Flextronics Institute.
- Currently active

Porting virtIO to POWERVM

- Adds value to the platform by bringing interesting new devices, like viftFS
- Evaluates how well virtio maps to different virtualization models
- Builds team skills around virtualization -> give back to the ecosystem



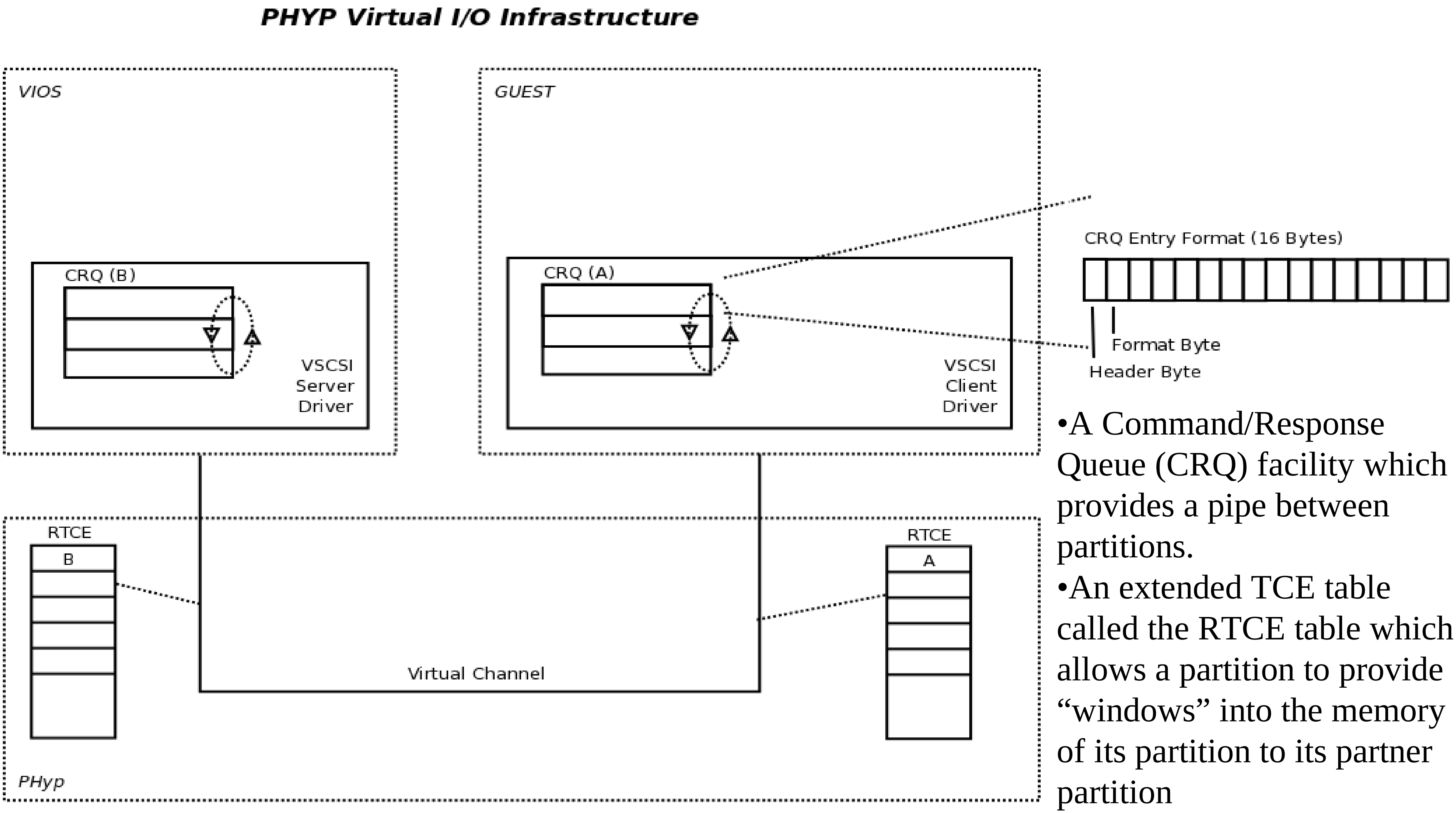
Virtual I/O on POWERVM



- **Virtual I/O Architecture**
  - **Mix of virtualized and/or real devices**
  - **Multiple VIO Servers* supported**
- **Virtual SCSI**
  - **Virtual SCSI, Fibre Channel, and DVD**
  - **Logical and physical volume virtual disks**
  - **Multi-path and redundancy options**

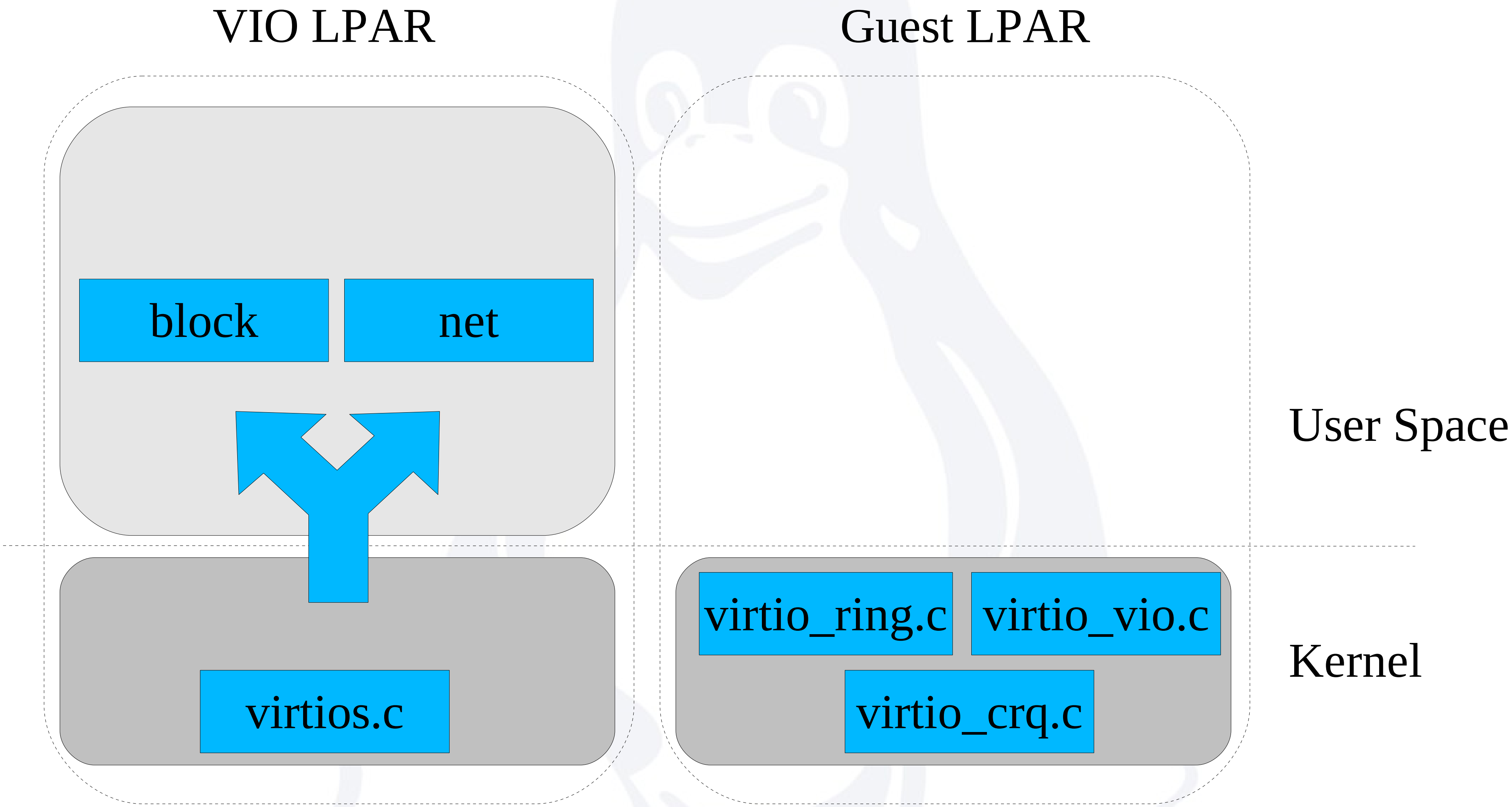
- **Benefits**
  - **Fewer adapters, I/O drawers, and ports**
  - **Improved speed to deployment**
- **Virtual Ethernet**
  - **VLAN and link aggregation support**
  - **LPAR-to-LPAR virtual LANs**
  - **Shared Ethernet adapter failover**

- **Virtual Fibre Channel**
  - **Utilizes N-Port ID Virtualization**
  - **Simplifies storage managment**



- A Command/Response Queue (CRQ) facility which provides a pipe between partitions.
- An extended TCE table called the RTCE table which allows a partition to provide “windows” into the memory of its partition to its partner partition
- Remote DMA services that allow a server partition to transfer data to a partner partition’s memory via the RTCE table window panes.

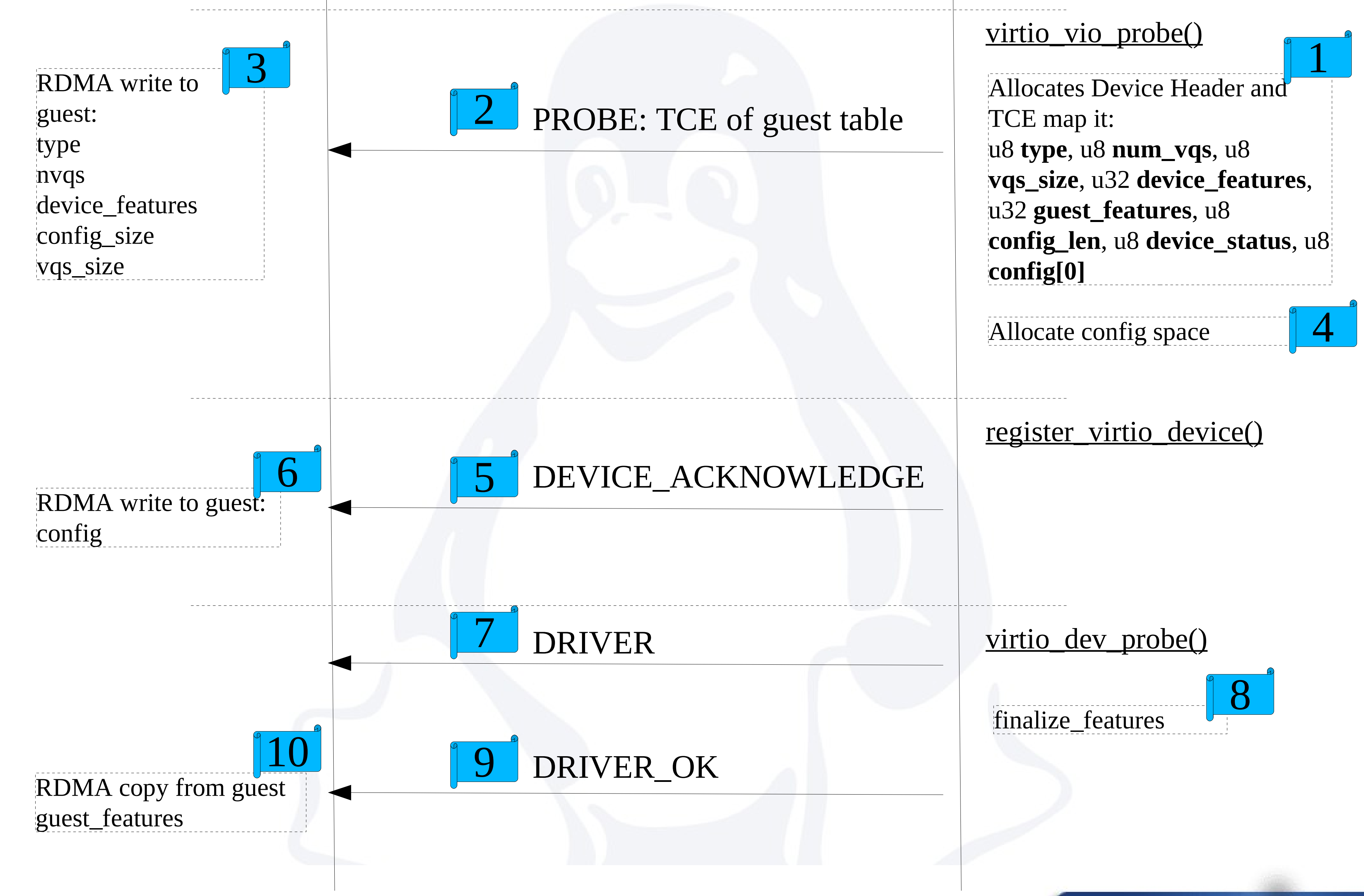
Implementation Details



VIO LPAR

Device Configuration

Guest LPAR



virtio_vio_probe()

1 Allocates Device Header and TCE map it:
u8 **type**, u8 **num_vqs**, u8 **vqs_size**, u32 **device_features**, u32 **guest_features**, u8 **config_len**, u8 **device_status**, u8 **config[0]**

2 PROBE: TCE of guest table

3 RDMA write to guest:
type
nvqs
device_features
config_size
vqs_size

4 Allocate config space

register_virtio_device()

5 DEVICE_ACKNOWLEDGE

6 RDMA write to guest:
config

virtio_dev_probe()

7 DRIVER

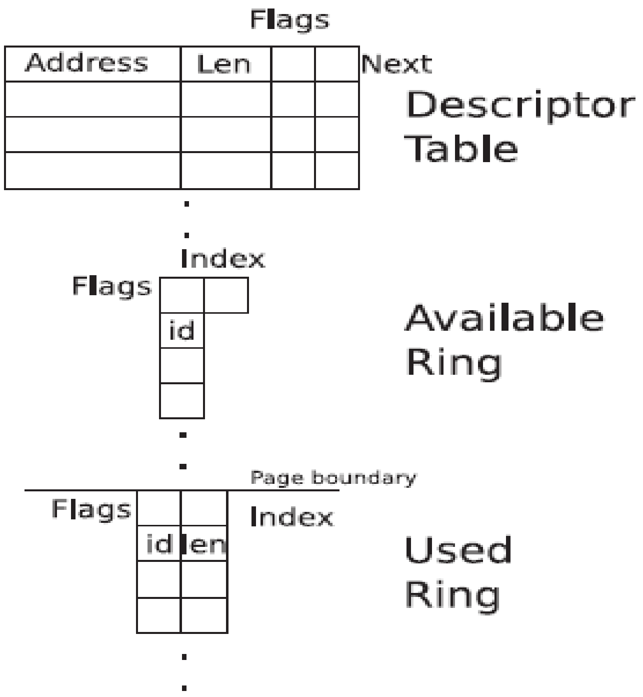
8 finalize_features

9 DRIVER_OK

10 RDMA copy from guest
guest_features

Virtqueues (plan)

- find_vqs
  - Expose TCEs for Descriptor Table, Available Ring and Used Ring
- Re-use vring
  - Hook HCALL_SEND_CRQ to vq.notify() - which is called by virtqueue_kick
    - Should cause the host to RDMA copy-in Descriptor Table and Available Ring
  - vring_desc.addr should hold TCEs, not Guest Physicals (u64 is fine, changing semantics only)
  - vring's add_buff should replace sg_phys() to sg_dma_address()
  - vring's detach should dma_unmap_sg() on each freed descriptor



References

- Power Architecture Platform Requirements (PAPR)
  - www.power.org
- “virtio: Towards a De-Facto Standard For Virtual I/O Devices”, Rusty Russel
- Virtio PCI Card Specication v0.8.8 DRAFT, Rusty Russel
- Kernel source tree