Linux Remote
&
Decouple RPMsg and Remoteproc

Etsam Anjum

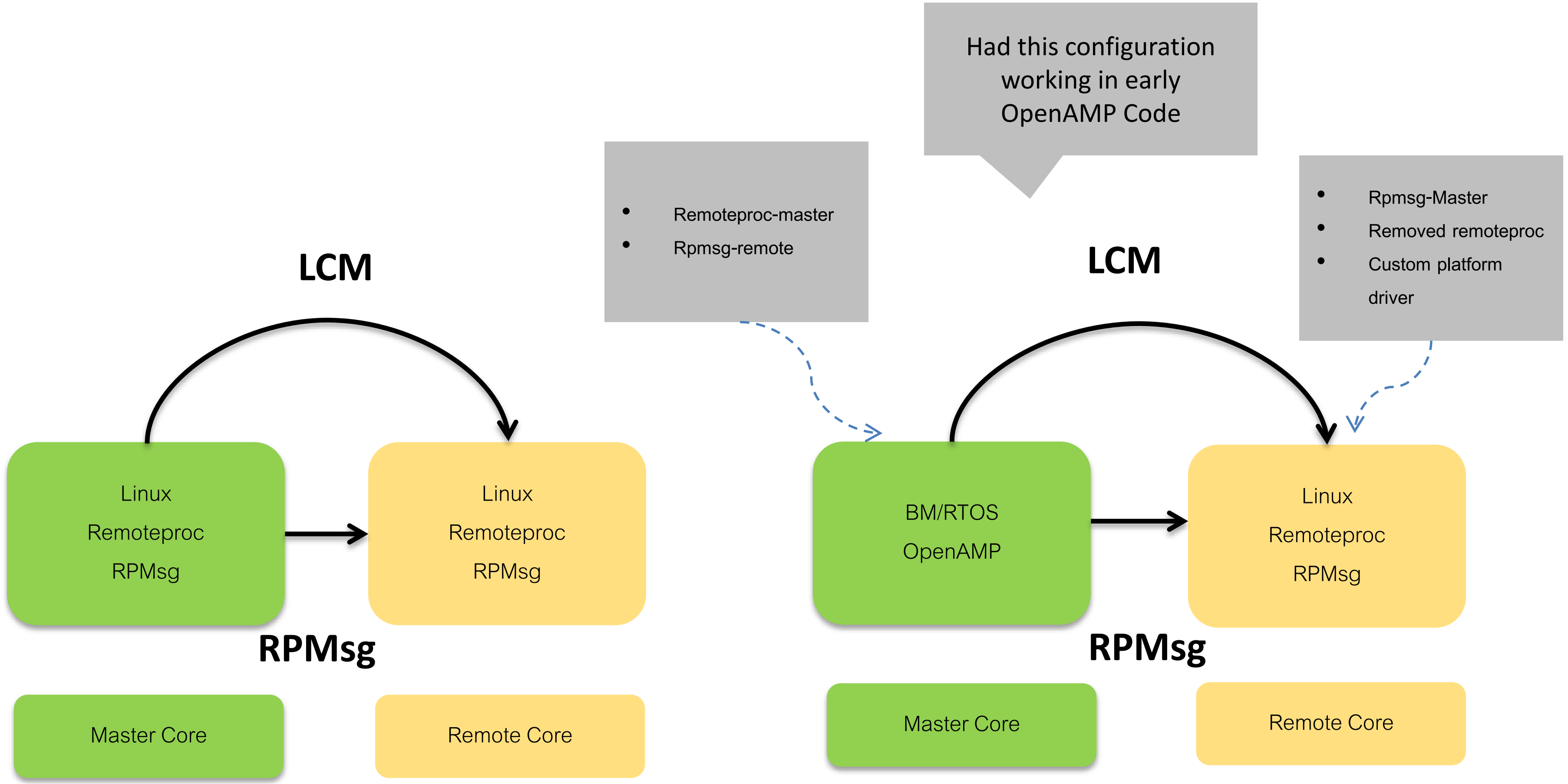
February 9, 2017

Two Topics

- Linux Remote
- Decouple RPMsg from Remoteproc

LINUX REMOTE

Configurations Not Possible Today



Had this configuration working in early OpenAMP Code

- Remoteproc-master
- Rpmsg-remote

- Rpmsg-Master
- Removed remoteproc
- Custom platform driver

LCM

Linux
Remoteproc
RPMsg

Linux
Remoteproc
RPMsg

RPMsg

Master Core

Remote Core

LCM

BM/RTOS
OpenAMP

Linux
Remoteproc
RPMsg

RPMsg

Master Core

Remote Core

Assumptions

- Remoteproc + RPMsg
- VirtIO (virtqueue) is a shared memory transport layer

RPMsg

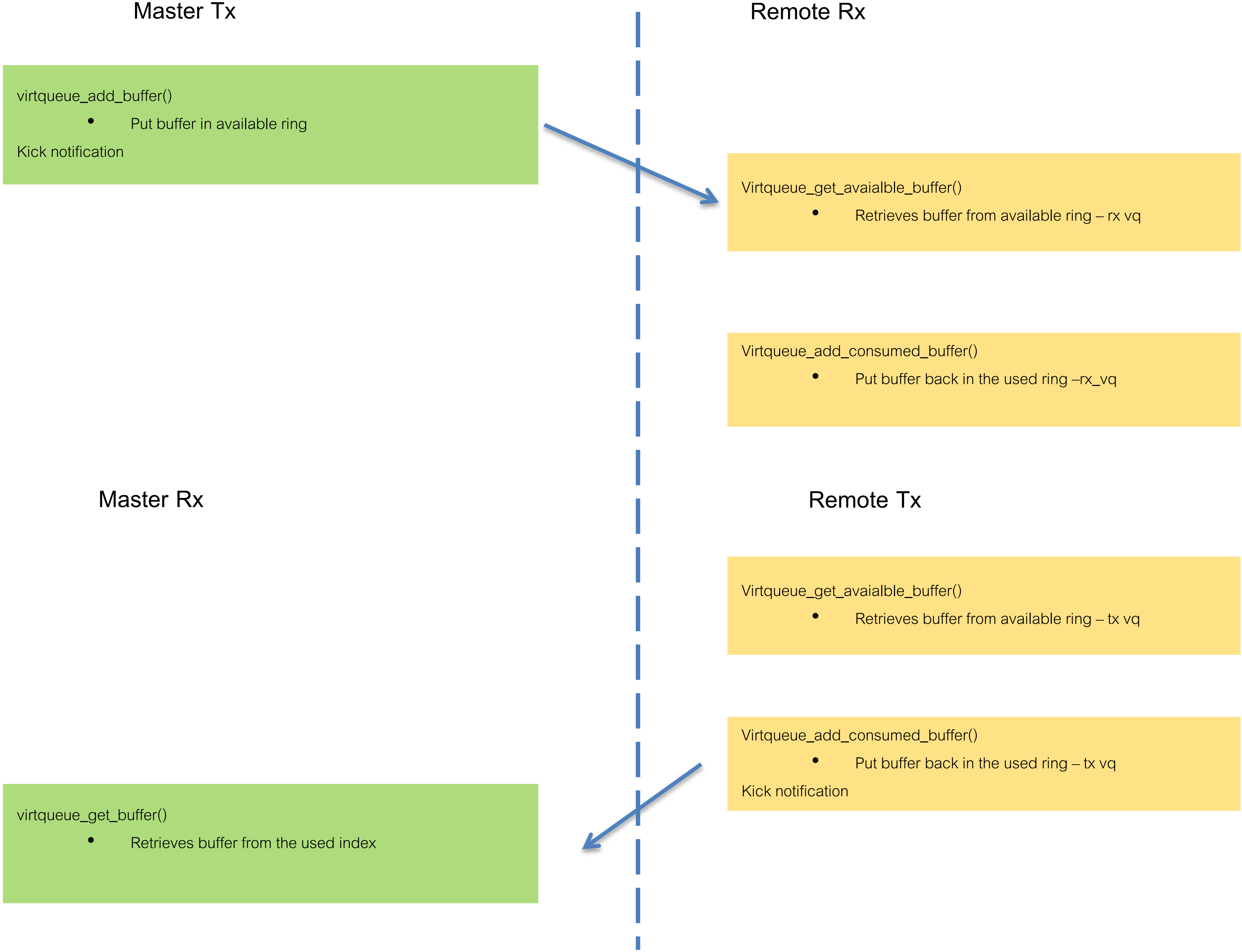
- Only has implementation for Master
  - Master Characteristics
    - Shared Memory Management
      - Creates DMA memory pool for shared memory
      - Populates buffers in the virtqueues, for remote Tx virtqueue also
    - Protocol
      - Triggers startup sequence
      - Features acknowledgement & status reporting - using Virtio device status field
      - Name service handling
- What if Linux is present on remote side – No support
- It's a VirtIO problem too
  - No code for data I/O from remote (backend) context

Solution - 1

- Enhance the RPMsg bus driver to provide remote functionality
  - Take execution path based on the role (master or remote)
  - Use build or runtime option to control the behavior
  - Master’s execution path will stay intact

Shared Memory Management

- Remote – no shared memory initialization and filling of virtqueues
- Virtqueue APIs for remote data I/O
  - `virtqueue_get_available_buffer`
    - Retrieve buffer from the available ring (both tx & rx vq)
  - `virtqueue_add_consumed_buffer`
    - Put buffer into virtqueue used ring



Master Tx

virtqueue_add_buffer()

- Put buffer in available ring

Kick notification

Remote Rx

Virtqueue_get_avaialble_buffer()

- Retrieves buffer from available ring – rx vq

Virtqueue_add_consumed_buffer()

- Put buffer back in the used ring –rx_vq

Master Rx

Remote Tx

Virtqueue_get_avaialble_buffer()

- Retrieves buffer from available ring – tx vq

Virtqueue_add_consumed_buffer()

- Put buffer back in the used ring – tx vq

Kick notification

virtqueue_get_buffer()

- Retrieves buffer from the used index

Protocol Handling

- Add code to handle remote side
  - Logic to track status reporting and take appropriate action
  - Handle startup interrupt – interrupt on TX virtqueue
  - Name service announcement
    - Code is present
    - Need to trigger it at right point
      - First Tx callback
      - Status is *DRIVER_OK*

Cont’d

- Pros
  - Reference design available in OpenAMP
- Cons
  - May not scale well for standalone RPMsg and peer to peer model

Solution - 2

- No *strict* notion of master and remote
- Let each side manage its Tx buffers
  - RPMSG driver
    - Both sides will manage shared memory
    - Each side manages its Tx virtqueue
  - VirtIO
    - Same set of APIs for each participating context
      - Not available today
      - virtqueue_add_buffer
        - Increments available index
        - Remote is supposed to get buffer from available index
      - virtqueue_get_buffer
        - Receives buffer from used index
        - Remote is supposed to put it there

Protocol Handling

- Status reporting
  - Cannot change much here
  - VirtIO configuration space requirement
  - One side will need to act as sort of *Master and Remote*
    - *Client server, initiator*
- Name Service announcement
  - Can be prevented with static channel creation feature
  - Enable NS advertising and handling in RPMsg stack
    - Code traces are present
- Trigger Interrupt
  - Can be eliminated, use status reporting
  - Active communication only after *Driver Ok* status

Cont’d

- Pros
  - Scalable to standalone RPMsg & peer to peer model
- Cons
  - Does not conform to VirtIO usage model
  - Backwards compatibility can be an issue

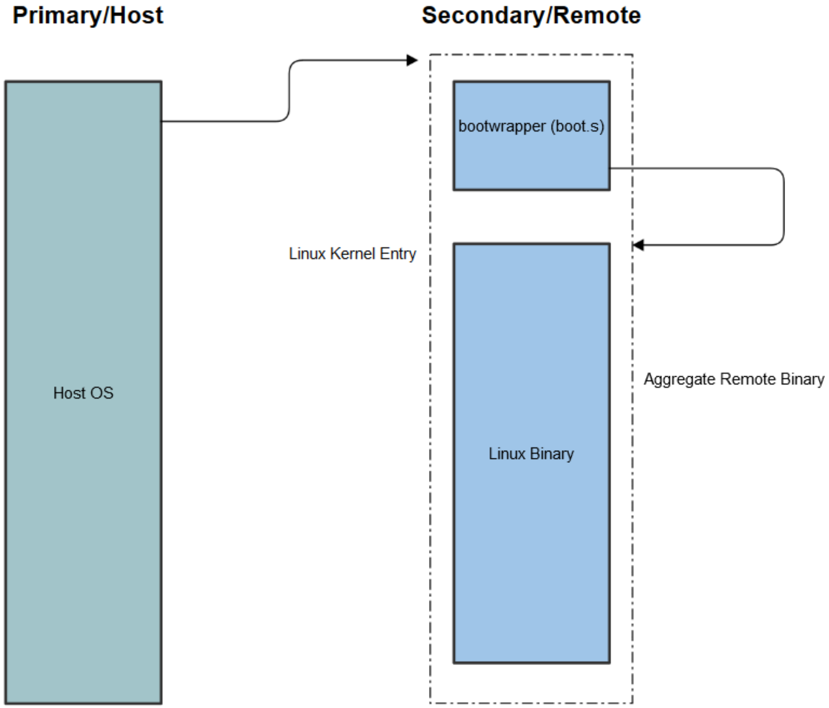
Remoteproc

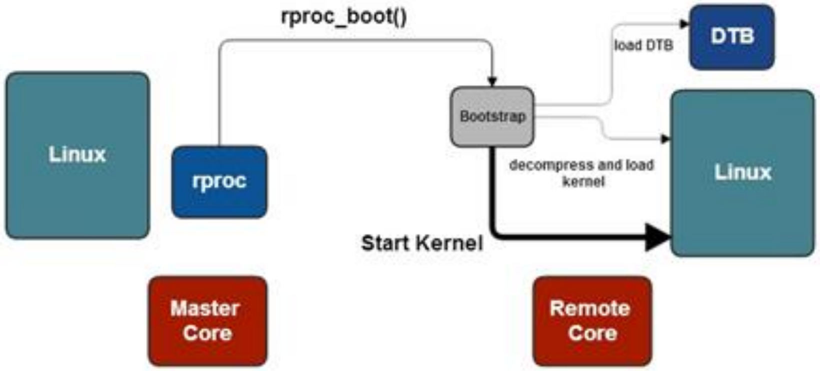
- What are the use cases:
  - BM/RTOS on safety critical processor controlling Linux Life Cycle
  - Linux booting another Linux?
    - inter guest communication, no remoteproc
- Cannot remove the concept of master and remote
- Master has distinct features which cannot be removed or absorbed in remote
  - Parsing the image
  - Carving out resources
  - Loading image & booting CPU

Remote’s Remoteproc

- Most of the required code is present
  - Resource table parsing
  - Just use resources – no carve out
  - Control execution path using build or runtime option
- Boot strap Linux
- Resource table sharing

Bootstrap Linux Image





Share Resource Table

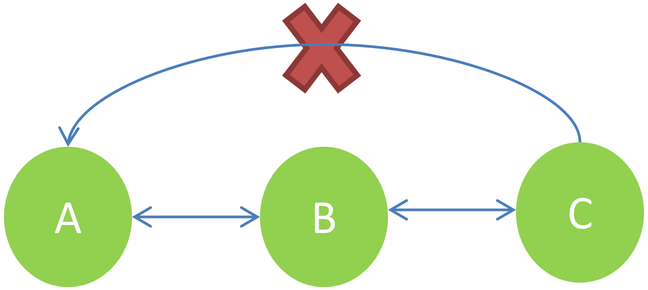
- Make it part of bootstrap ELF
  - Master can access it without any issue
  - How to provide access to remote ?
    - Fix memory location of resource table at compile time
    - Place resource table at that location in bootstrap ELF - using linker script and section attributes
      - Master can still get it from ELF
        - Code is reused
      - User will provide the address in remote Linux DTS
        - Need to provide address at different places
      - Platform driver on Linux remote side
  - Can boostrap pass address of resource table to Linux remote
    - Cannot relocate the resource table
    - Suggestions?

Suggestions

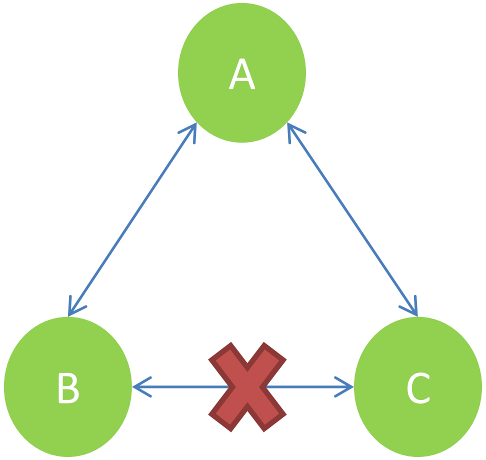
- *Suggestions from Bjorn and Tomas*
  - *Patch the DTB from bootstrap code to provide the resource table address. User will not need to provide it in the device node.*
  - *Include libfdt support in the bootstrap to parse DTB and handle other image types*

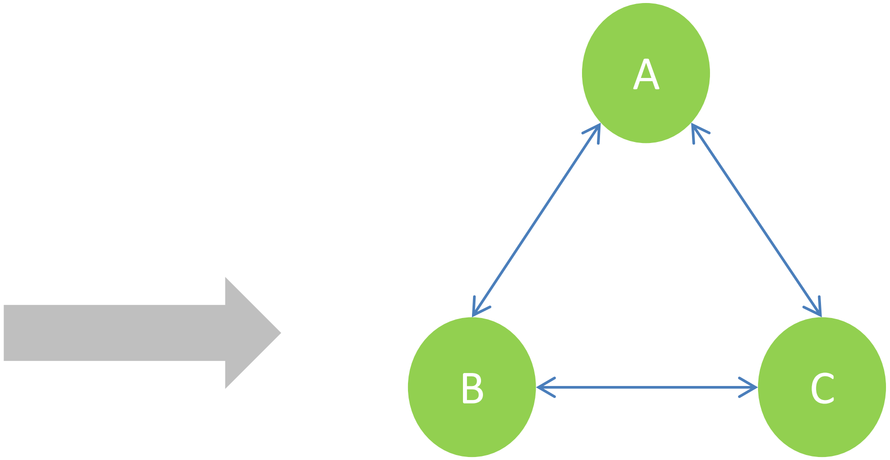
DECOUPLE RPMSG AND REMOTEPROC

Motivation



Peer to Peer communication is not possible



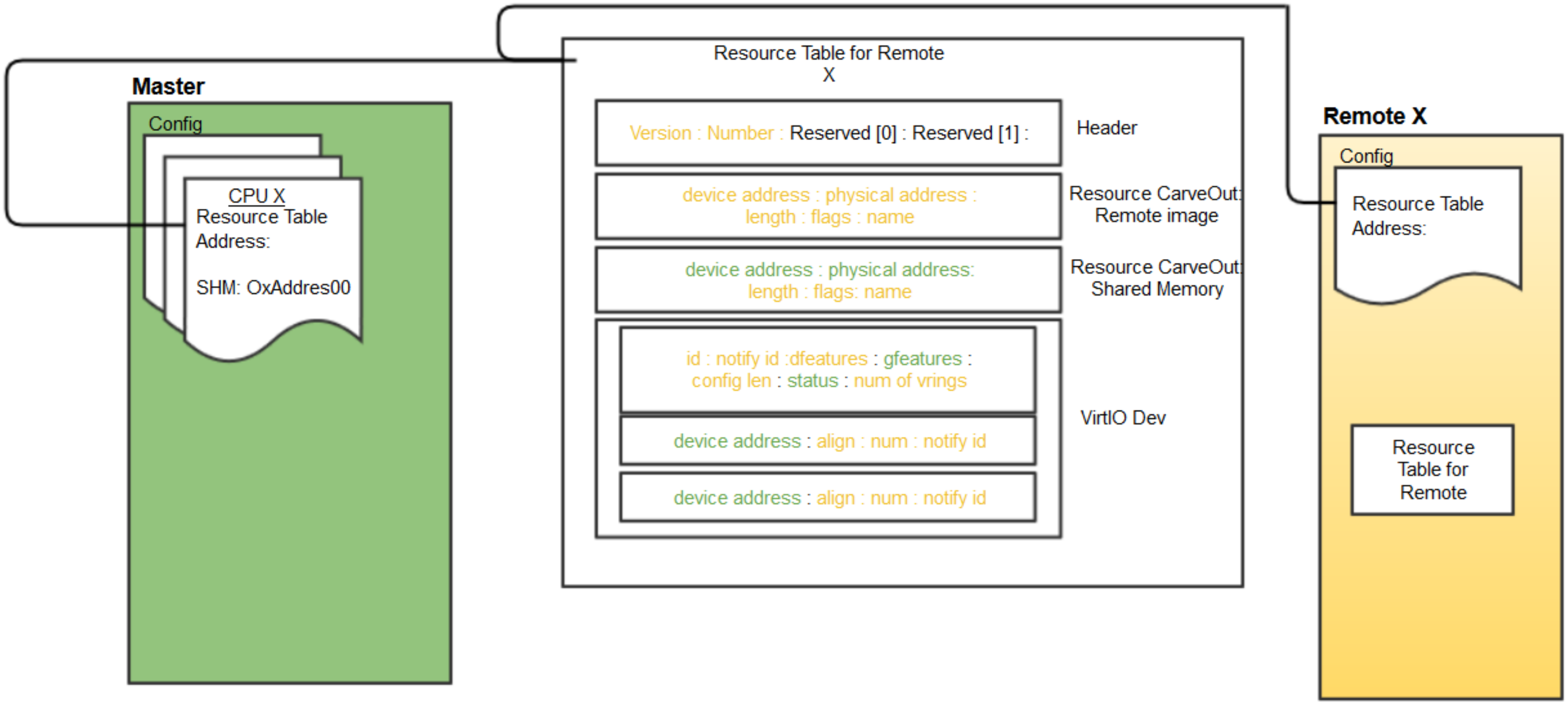


Boot loader will bring up images on all cores

Cont’d

- Information Sharing
- Protocol
- Buffer Management

Information Sharing



Master
Config
CPU X
Resource Table
Address:
SHM: OxAddres00
Resource Table for Remote
X
Version : Number : Reserved [0] : Reserved [1] :
Header
device address : physical address :
length : flags : name
Resource CarveOut:
Remote image
device address : physical address:
length : flags: name
Resource CarveOut:
Shared Memory
id : notify id :dfeatures : gfeatures :
config len : status : num of vrings
device address : align : num : notify id
device address : align : num : notify id
VirtIO Dev
Remote X
Config
Resource Table
Address:
Resource
Table for
Remote

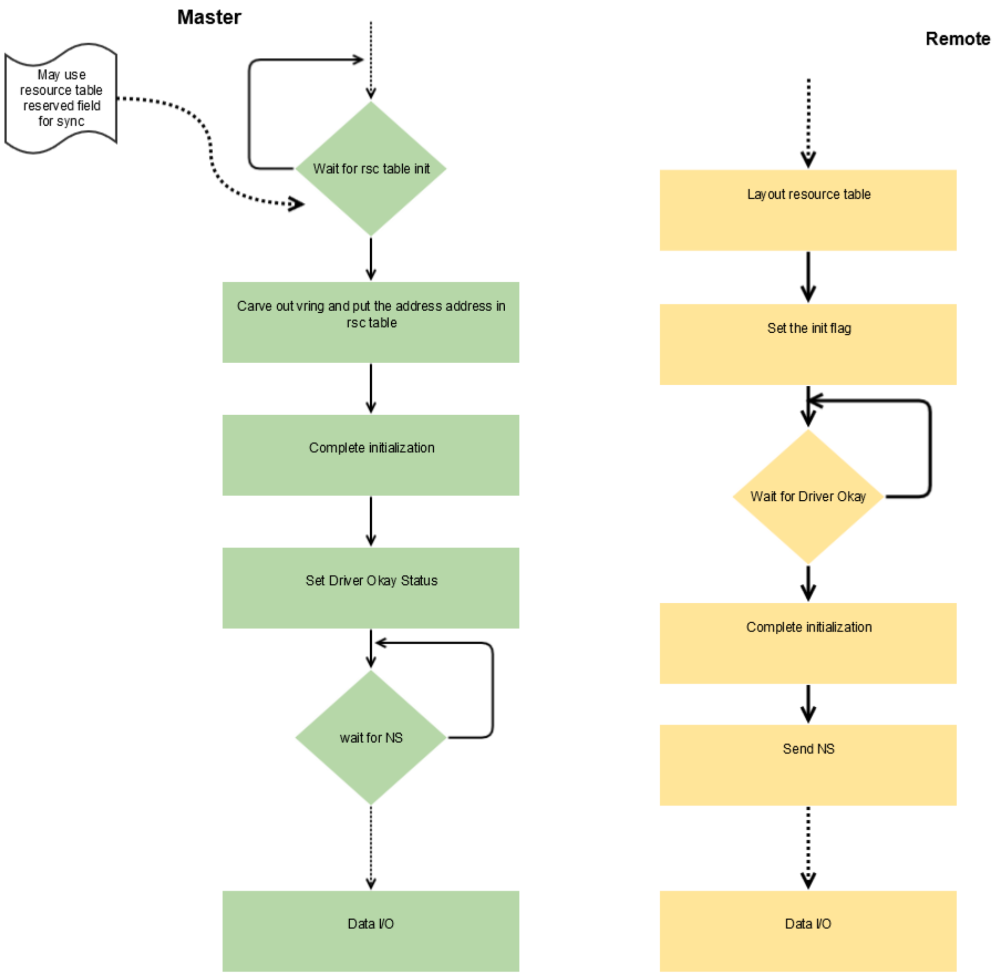
Cont’d

- Same information is required even without remoteproc/resource table
- How to provide?
  - Fix memory region at compile time
  - Share it with the software contexts involved (using dts, header file)
- Layout of Shared Memory?
  - Keep it similar to resource table
    - No need to define new layout – *specs standpoint*
    - Some resource are not required - *Image carve out resource*
      - Resource table has dynamic size, determined by the header
    - May need new resource – *pass on shared memory info*

Information Initialization

Approach 1

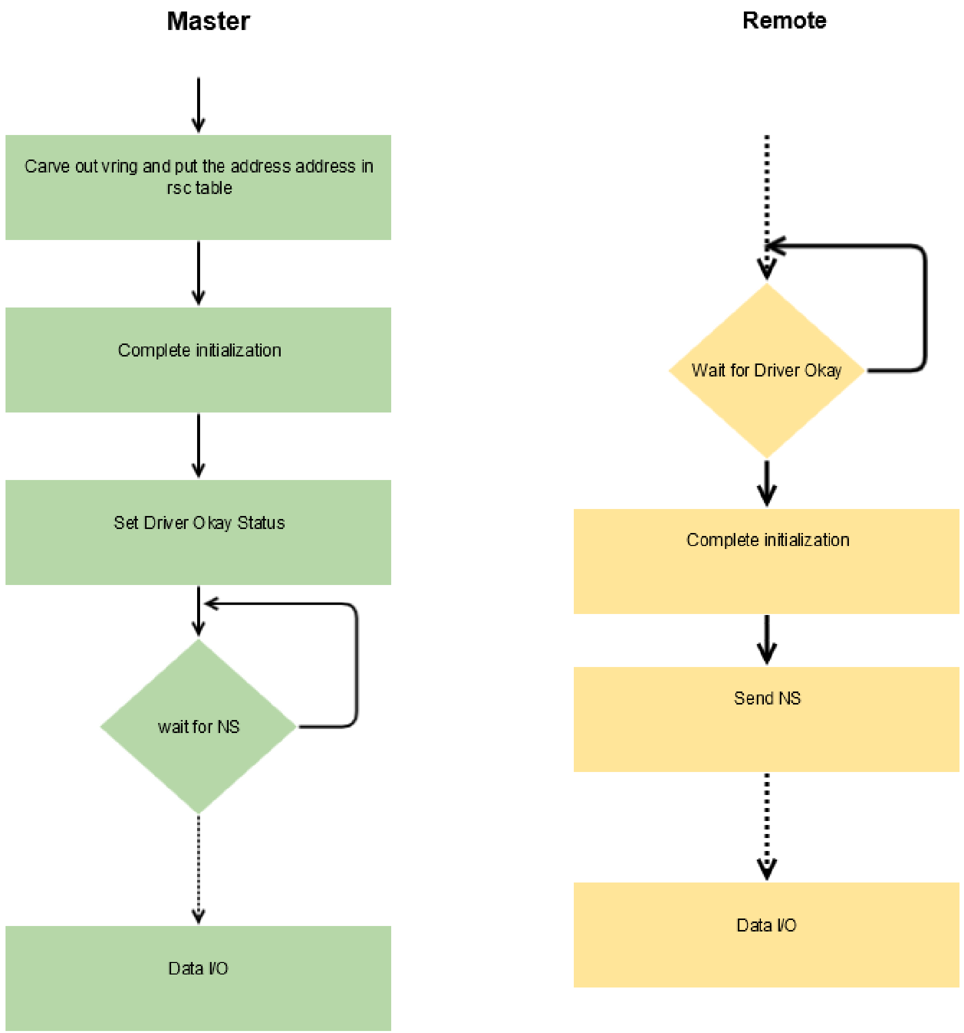
- Similar to remoteproc
  - Master waits for Remote to lay out resource table
  - Remote lays out Resource table, signal master to update resource table and wait for master's sign
  - Master setups all the resources(SHM, VirtQues) and updates resource table. Gives remote go ahead.
  - Remote uses resource table and sets up all the resources on its end.
  - Establish channel and start communication



Cont'd

Approach 2

- Resource table will be loaded by the boat loader
    - Will have initial values, previously provided by the remote
- Access synchronizatiom is not required at the start



Cont’d

- Master still provides the address of vring!
  - Master can carve out shared memory region from its own address space
  - Does not bind the allocation mechanisms in different software environments
- Participating contexts maintain address of vring
  - Vring must start at the same address

Suggestions

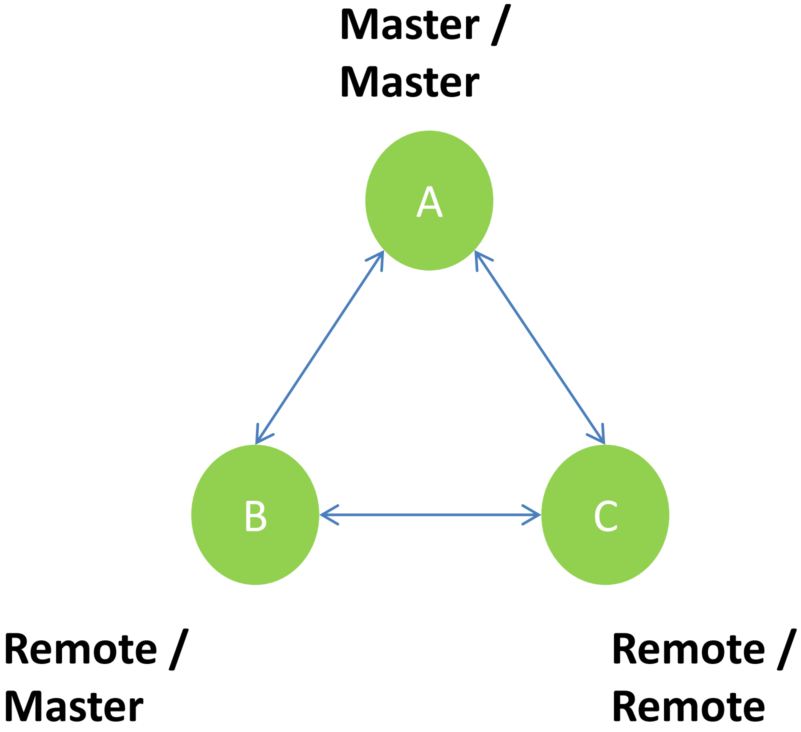
- Suggestions from Bjorn and Tomas
  - Initially enable information sharing without resource table in shared memory. May just replicate the information on both sides
  - The approach 1 may introduce deadlock during initialization

Implementation Perspective

- Impact on Linux drivers
  - virtio_remoteproc.c
    - Implements VirtIO device config ops
    - Share the code between remoteproc and rpmsg
  - Share the resource table parsing code between remoteproc and rpmsg
  - Add new resource in resource table to pass shared memory information [specially for buffers]
  - New platform driver

Peer to Peer Model

- Without remoteproc, Peer to Peer communication is possible
- RPMsg driver can still have master - slave architecture
- Consider the example where in
  - Three software contexts
  - Bootloader boots the images
  - Each communication link has its own resources, shared memory, IPIs
  - Every software environment is aware of these resources
  - A: Master for B and C
  - B: Remote for A, master for C
  - C : Remote for A, remote for B



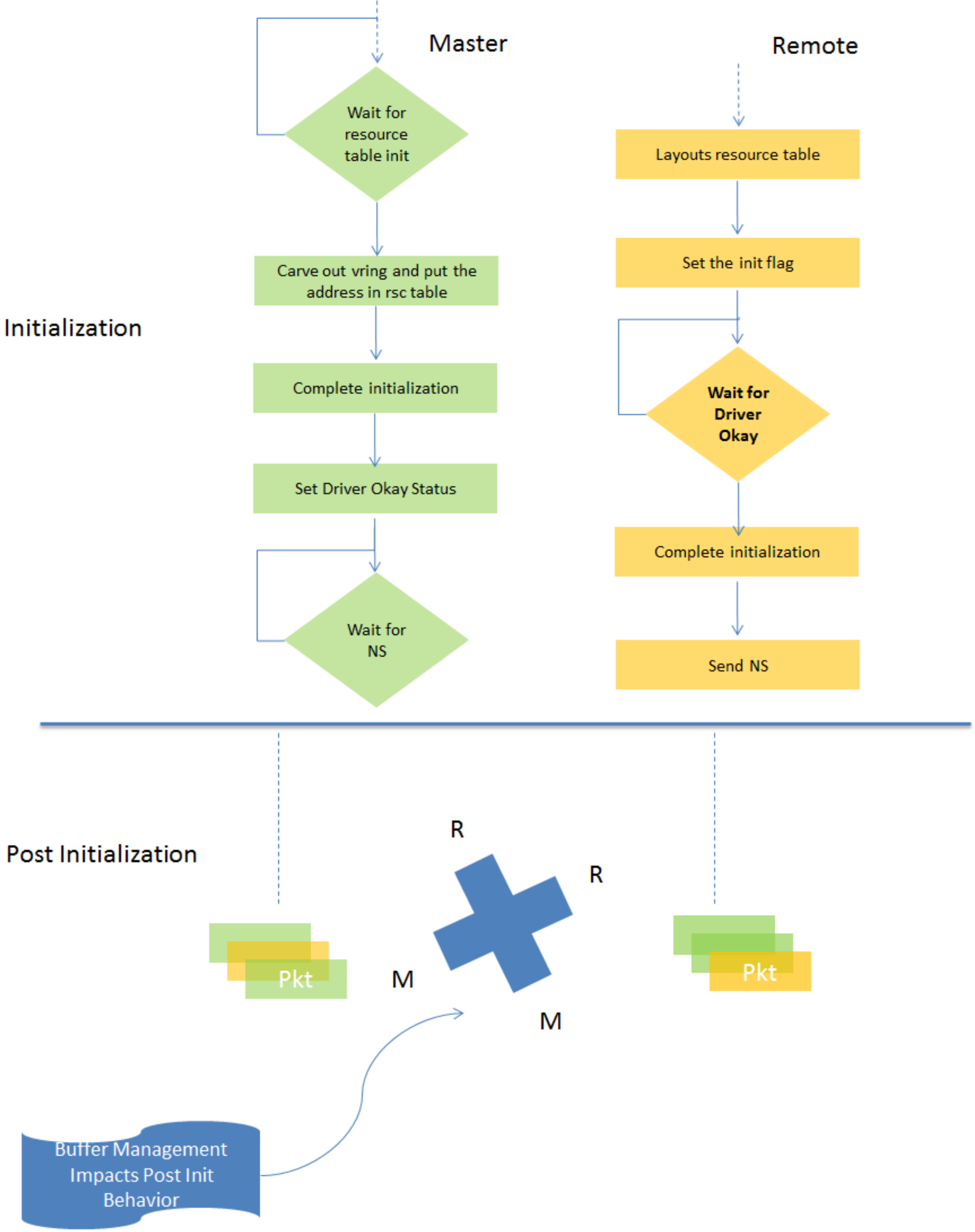
Master / Master

Remote / Master

Remote / Remote

Buffer Management

- Its an RPMSG issue
- Let the each side control its TX virtqueue completely
- Suppress the master/slave architecture
- More close “Peer to Peer”



Master

Remote

Initialization

Wait for resource table init

Carve out vring and put the address in rsc table

Complete initialization

Set Driver Okay Status

Wait for NS

Layouts resource table

Set the init flag

Wait for Driver Okay

Complete initialization

Send NS

Post Initialization

Buffer Management Impacts Post Init Behavior

Design

- Please see slides 6-11

Thank You