Embedded Linux Conference
Europe

Asymmetric Multiprocessing and Embedded Linux

Marek NOVAK, Dusan CERVENKA

October 24, 2017

THE LINUX FOUNDATION

Who are we?

- Marek NOVAK
  - Author/maintainer of RPMsg-Lite library
  - PhD student
- Dusan CERVENKA
  - Author/maintainer of eRPC library
- Both
  - Linux entousiasts
  - Working at NXP Semiconductors

Outline

Asymmetric Multiprocessing (AMP)

Remote Processor Messaging (RPMsg)
in Linux kernel

RPMsg-Lite – RPMsg for RTOS

Embedded Remote Procedure Call (eRPC)

Asymmetric Multiprocessing (AMP)

Motivation – why AMP? 1/2

- Thanks to AMP, your system can:
  - Be faster
  - Consume less power
  - Be safer
  - Be more secure

- How?

Motivation – why AMP? 2/2

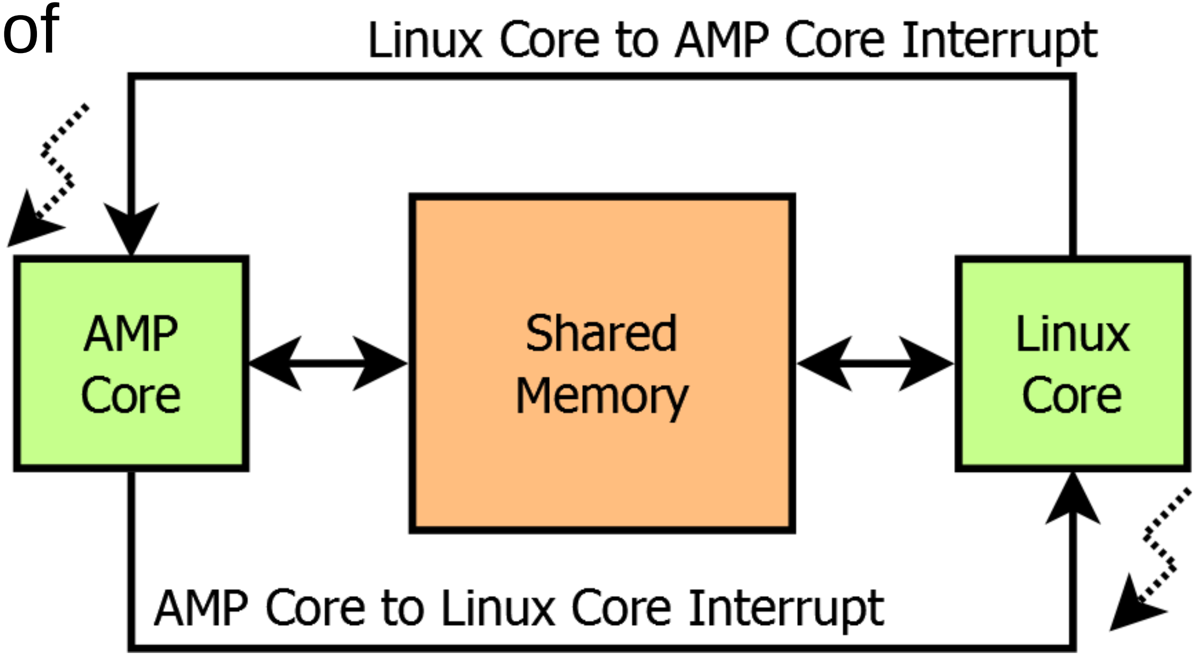
- Not all CPUs in the system are treated equally
- The cores can:
  - Run independently
  - Have different operating systems
  - Have different architecture, clock frequency
  - Be tailored for a specific task (DSP)

How does it work?

- There is a concept of master and slave

- Master manages shared memory

- Master may control slave’s life-cycle

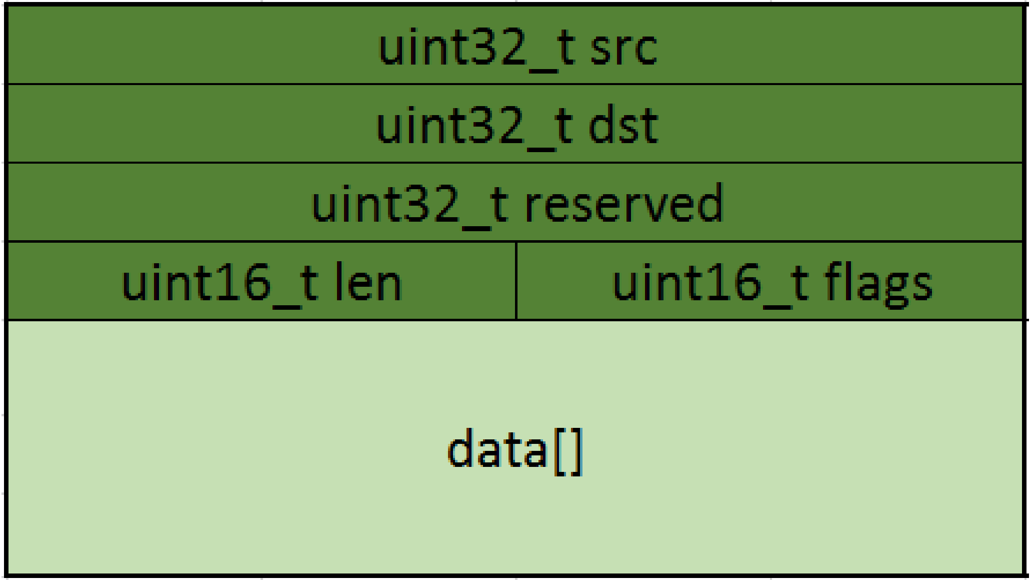

Linux Core to AMP Core Interrupt
AMP Core
Shared Memory
Linux Core
AMP Core to Linux Core Interrupt

Remote Processor Messaging (RPMsg) in Linux kernel

What is RPMsg?

- RPMsg defines a UDP-like header

- Only the header is strictly defined, there exist multiple transport mechanism



uint32_t src

uint32_t dst

uint32_t reserved

uint16_t len

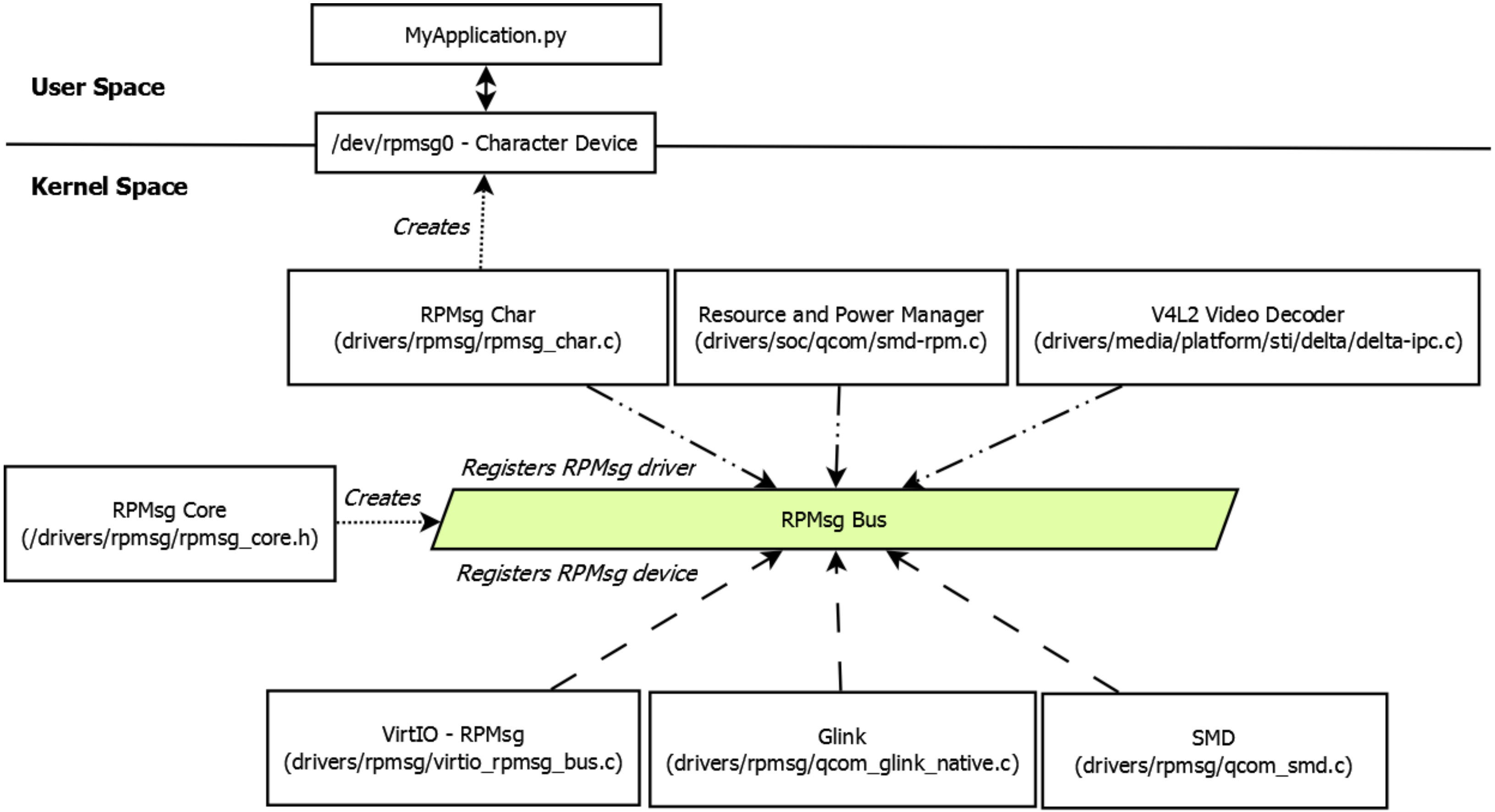
uint16_t flags

data[]

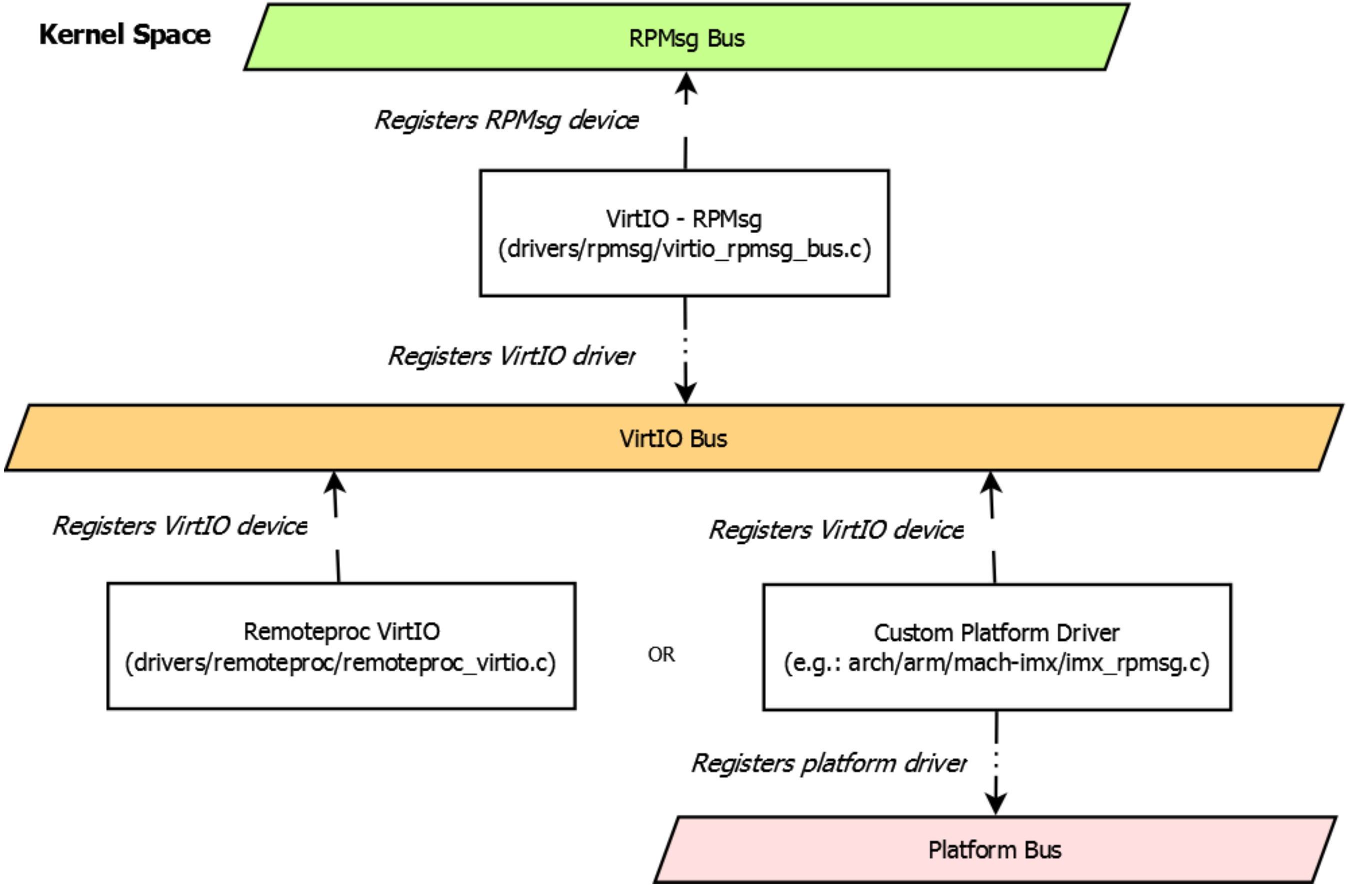
RPMsg in Linux - History

- RPMsg used to have only one transport layer based on VirtIO
- Initially maintained by Ohad Ben Cohen
- Now maintained by Bjorn Andersson (Linaro)
- New transport layers added – Glink and SMD
  - Mostly for Qualcomm platforms

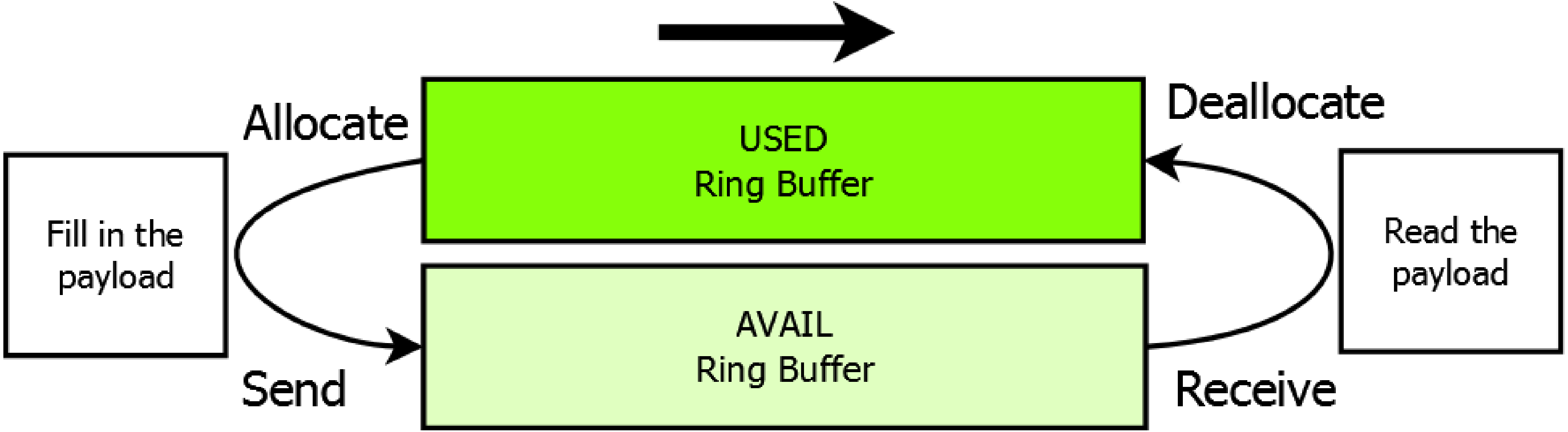
RPMsg in Linux – Current State 1/2


MyApplication.py
User Space
/dev/rpmsg0 - Character Device
Kernel Space
Creates
RPMsg Char
(drivers/rpmsg/rpmsg_char.c)
Resource and Power Manager
(drivers/soc/qcom/smd-rpm.c)
V4L2 Video Decoder
(drivers/media/platform/sti/delta/delta-ipc.c)
Registers RPMsg driver
RPMsg Core
(/drivers/rpmsg/rpmsg_core.h)
Creates
RPMsg Bus
Registers RPMsg device
VirtIO - RPMsg
(drivers/rpmsg/virtio_rpmsg_bus.c)
Glink
(drivers/rpmsg/qcom_glink_native.c)
SMD
(drivers/rpmsg/qcom_smd.c)

RPMsg in Linux – Current State 2/2

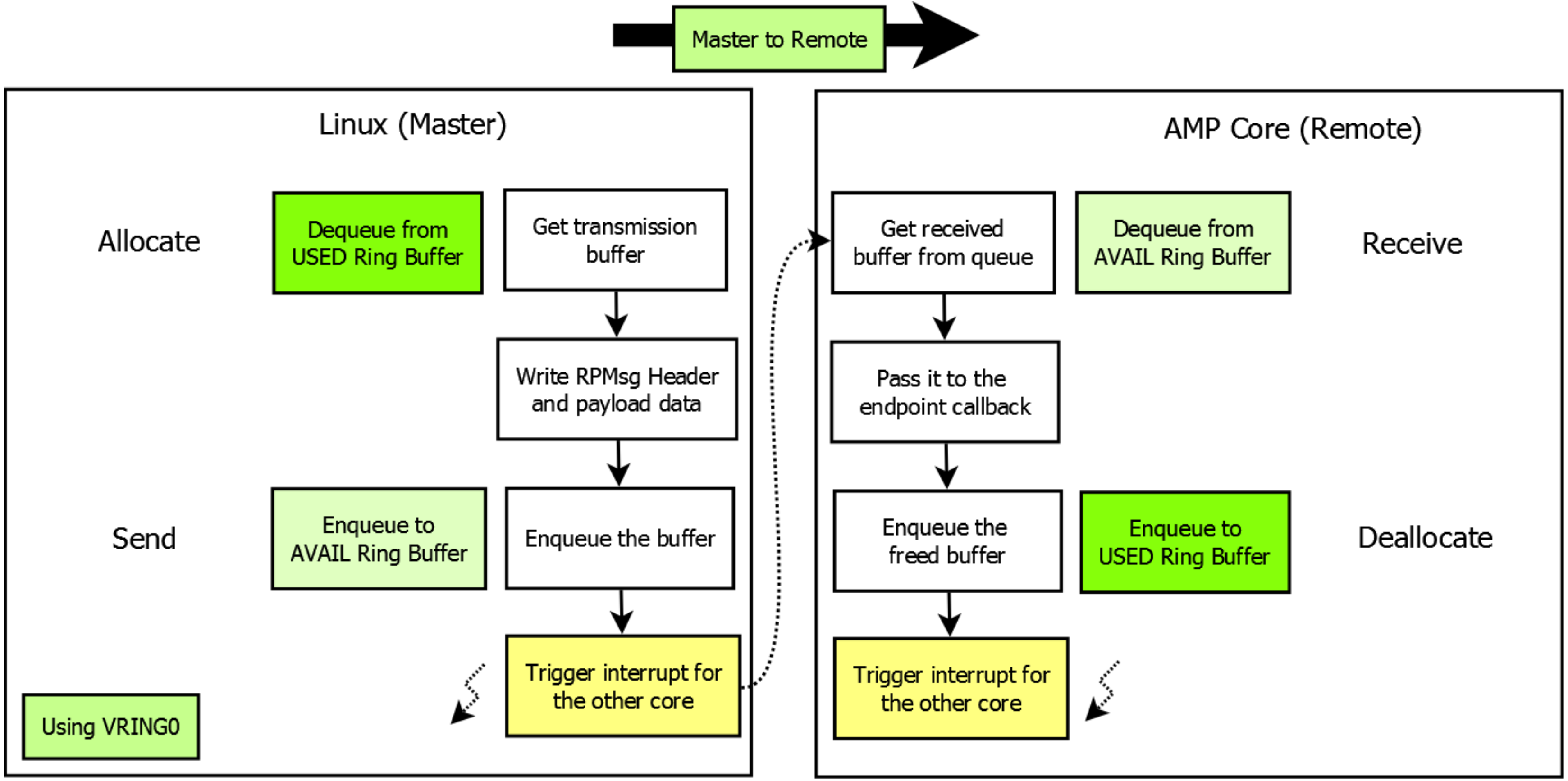


RPMsg/VirtIO 1/3



- Single writer single reader approach
- Allows for zero-copy
- 2 ring buffers for each direction

RPMsg/VirtIO 2/3



Master to Remote

Linux (Master)

Allocate

Dequeue from USED Ring Buffer

Get transmission buffer

Write RPMsg Header and payload data

Send

Enqueue to AVAIL Ring Buffer

Enqueue the buffer

Trigger interrupt for the other core

Using VRING0

AMP Core (Remote)

Get received buffer from queue

Dequeue from AVAIL Ring Buffer

Receive

Pass it to the endpoint callback

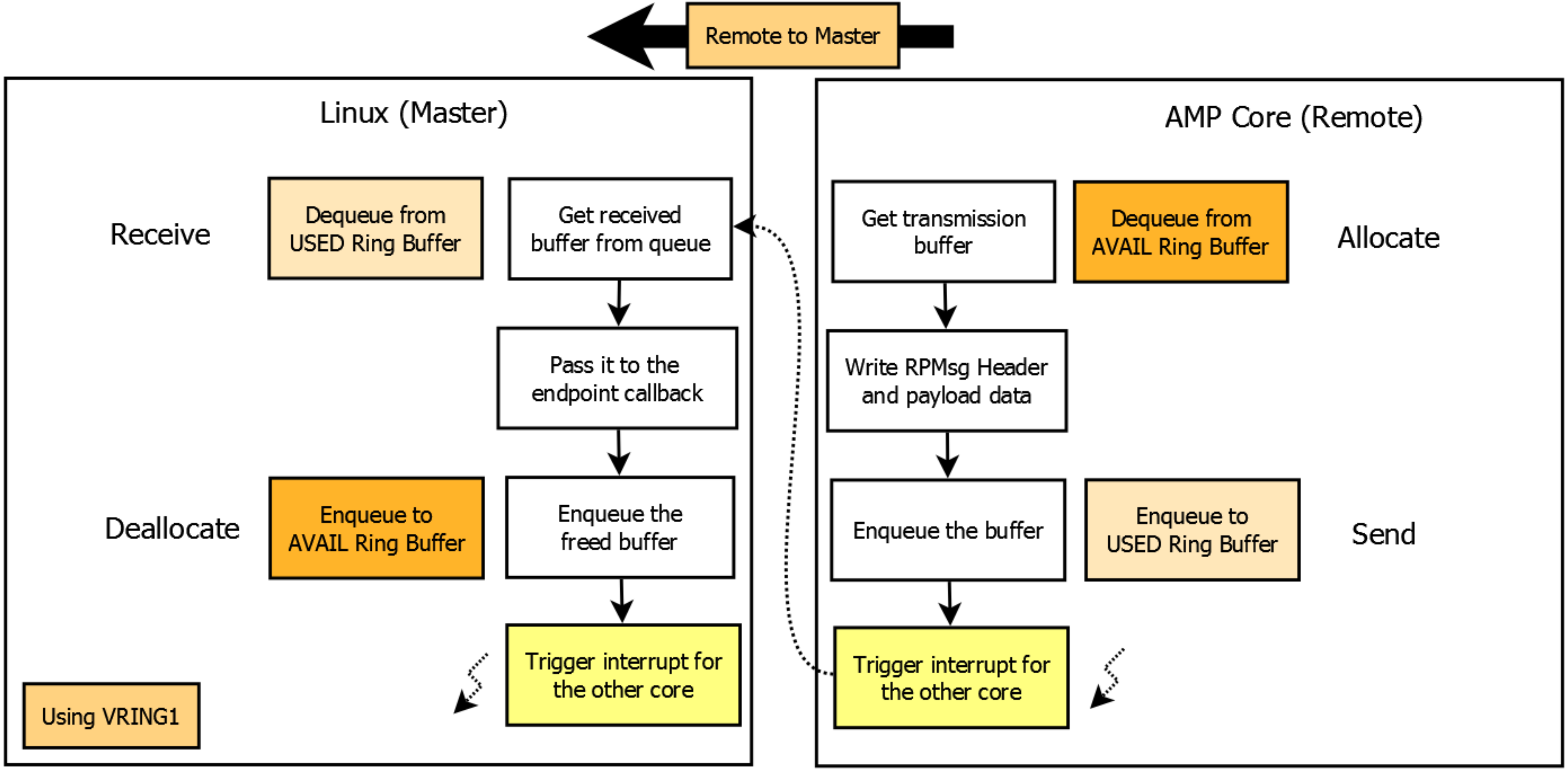
Enqueue the freed buffer

Enqueue to USED Ring Buffer

Deallocate

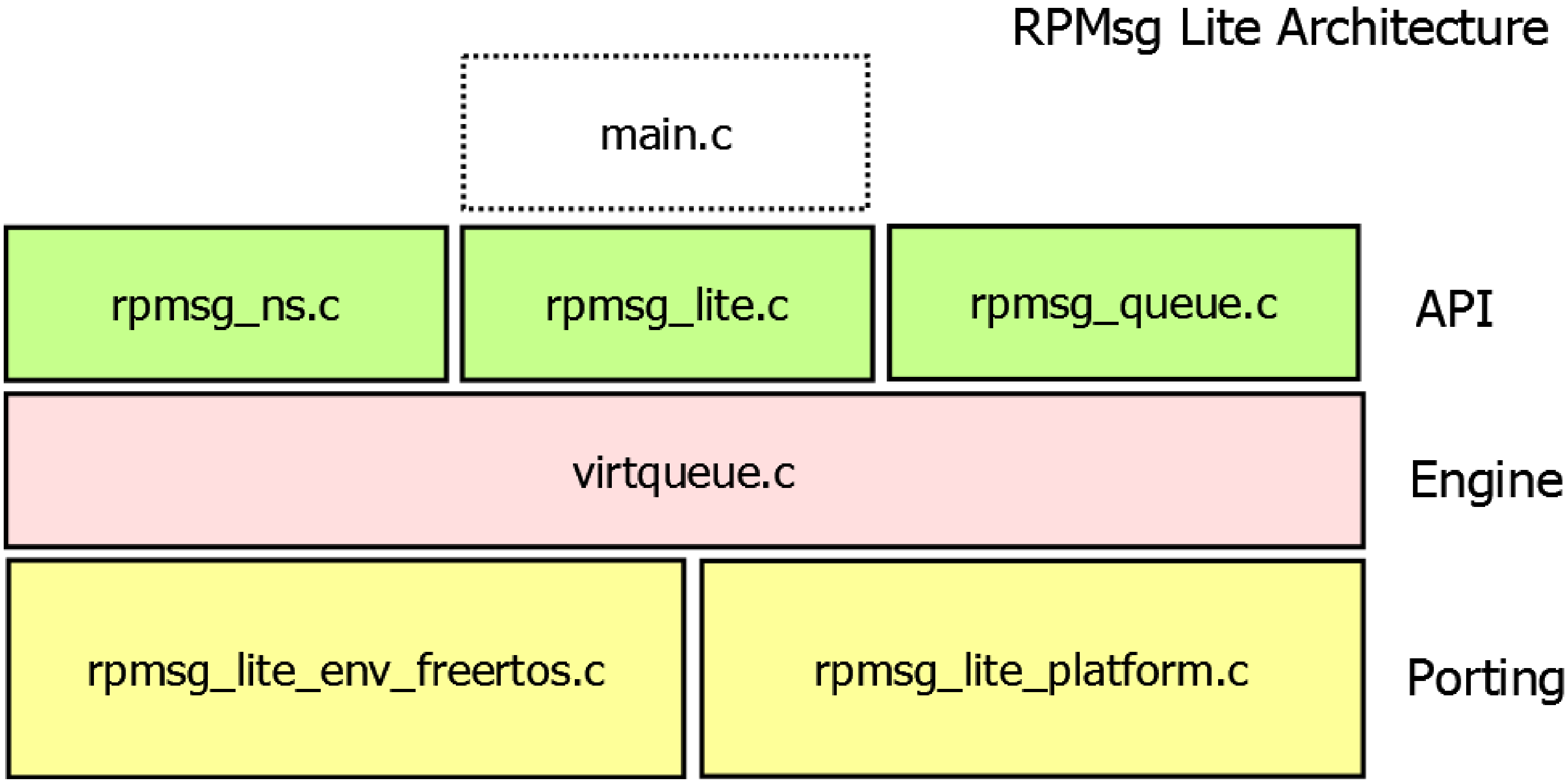
Trigger interrupt for the other core

RPMsg/VirtIO 3/3


Remote to Master
Linux (Master)
Receive
Dequeue from USED Ring Buffer
Get received buffer from queue
Pass it to the endpoint callback
Deallocate
Enqueue to AVAIL Ring Buffer
Enqueue the freed buffer
Trigger interrupt for the other core
Using VRING1
AMP Core (Remote)
Get transmission buffer
Dequeue from AVAIL Ring Buffer
Allocate
Write RPMsg Header and payload data
Enqueue the buffer
Enqueue to USED Ring Buffer
Send
Trigger interrupt for the other core

RPMsg-Lite – RPMsg for RTOS

Library Architecture 1/2



RPMsg Lite Architecture

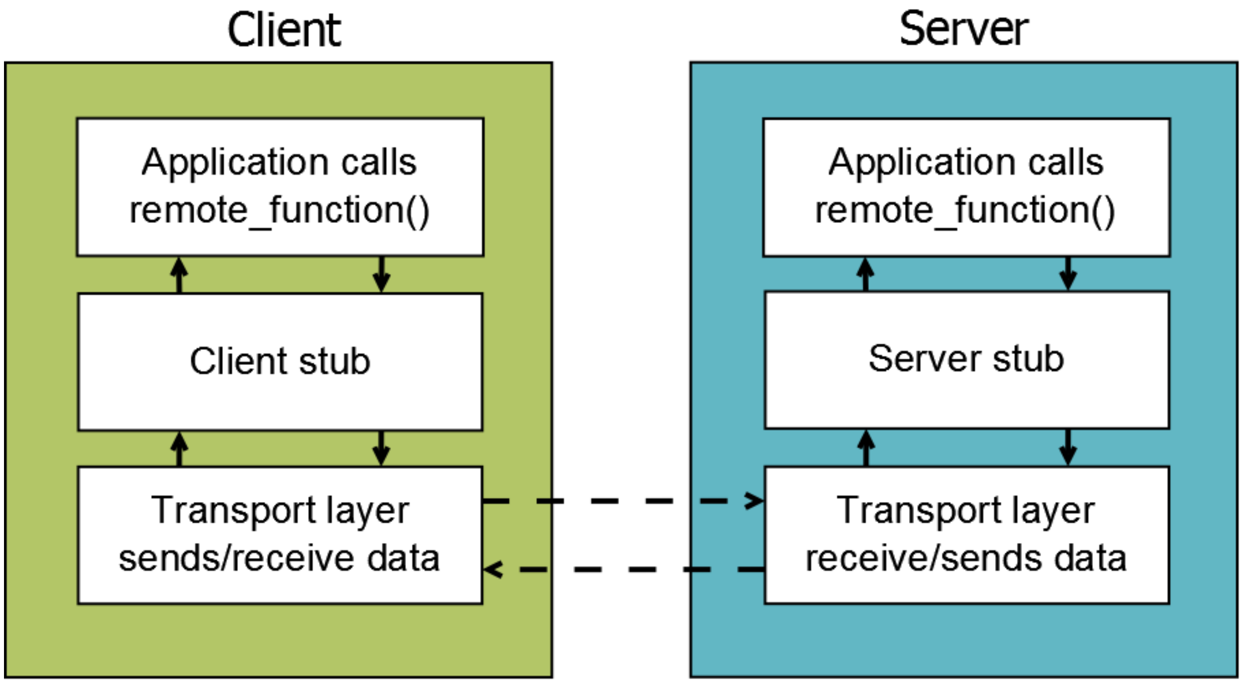
Library Architecture 2/2

- Environment (“RTOS”) porting layer
- Platform (“hardware”) porting layer
- Modular and simple
- BSD licensed
- FreeRTOS port (done) and Zephyr port (TBD)
- Github: https://github.com/NXPmicro/rpmsg-lite

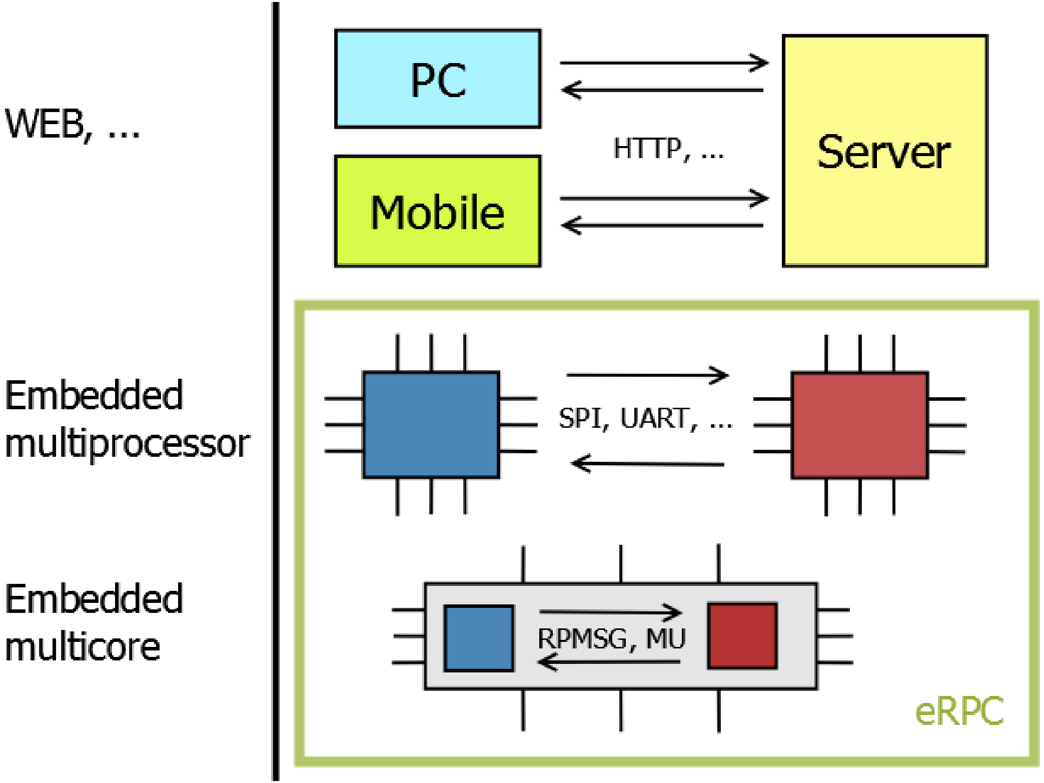
Embedded Remote Procedure Call (eRPC)

- What is RPC?
- Why eRPC?
- How to use eRPC?

What is RPC?


Client
Server
Application calls remote_function()
Client stub
Transport layer sends/receive data
Application calls remote_function()
Server stub
Transport layer receive/sends data

Why eRPC? 1/2



Other RPC:

- Apache Thrift
- Microsoft RPC
- Google RPC
- JSON-RPC
- …

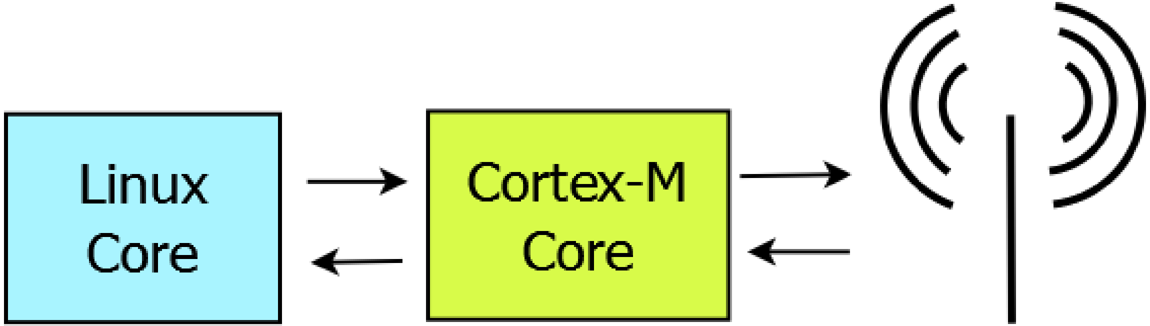
Why eRPC? 2/2

- Small code size
- Programing languages: C/C++, Python, … ?
- APIs style defined by user.
- Generated stub code
- Easy to port: BM, FreeRTOS, Zephyr(TBD)
- Modular and simple
- BSD-3: https://github.com/EmbeddedRPC/erpc

How to use eRPC? 1/3

Dual core device – Linux core + Cortext-M core

- Main core – Multimedia applications, web server, …
- Second core – communicating through radio



How to use eRPC? 2/3

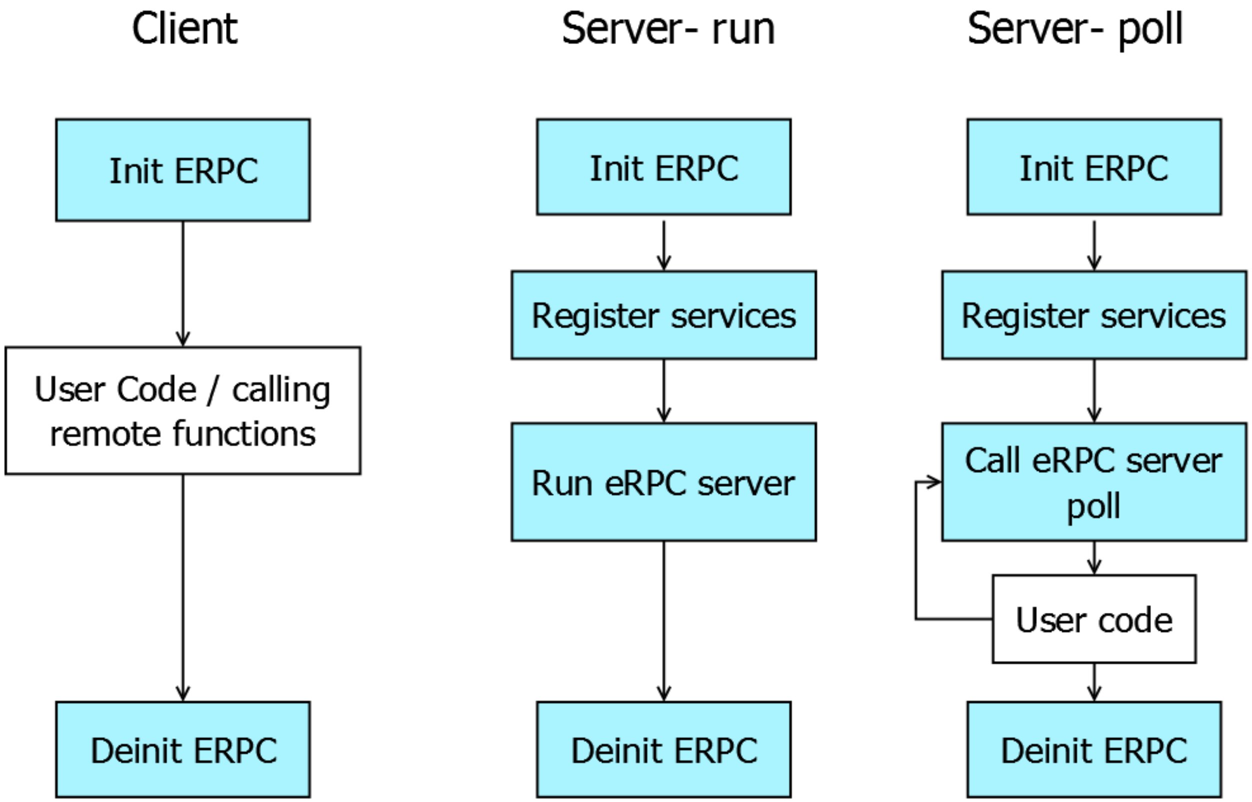
- Interface Definition Language (IDL):

```
interface Radio {
    send_packet(uint16 addr, uint8 dataLength,
                binary data @length(dataLength)) -> bool
}
```

- Generated declaration:

```
bool send_packet(uint16_t addr, uint8_t dataLength,
                 const uint8_t* data);
```

How to use eRPC? 3/3



Client

Init ERPC

User Code / calling remote functions

Deinit ERPC

Server- run

Init ERPC

Register services

Run eRPC server

Deinit ERPC

Server- poll

Init ERPC

Register services

Call eRPC server poll

User code

Deinit ERPC

Q & A

- RPMsg-Lite: https://github.com/NXPmicro/rpmsg-lite
- Marek NOVAK: marek.novak@nxp.com
- eRPC: https://github.com/EmbeddedRPC/erpc
- Dusan CERVENKA: dusan.cervenka@nxp.com

Embedded Linux Conference

Europe

THE LINUX FOUNDATION