RPMSG PROTOCOL STANDARDIZATION – KICK OFF

PETR LUKAS, MAREK NOVAK
MCU SW TEAM
JUN 02, 2016

SECURE CONNECTIONS FOR A SMARTER WORLD

RPMSG protocol standardization

- Current protocol is defined by implementations:
  - RPMSG Kernel module
  - RPMSG OpenAMP
  - Silicon vendors implementations (NXP, TI)

- New implementations needs standard!
  - OpenAMP 2016.10 lib-metal and move to UserSpace
  - NXP creating lightweight implementation for BM/RTOS to BM use case

- Tasks
  - **Describe current status protocol !**
  - Need to separate implementation and protocol definition
  - Need to understand future extendibility of protocol

What we need to describe

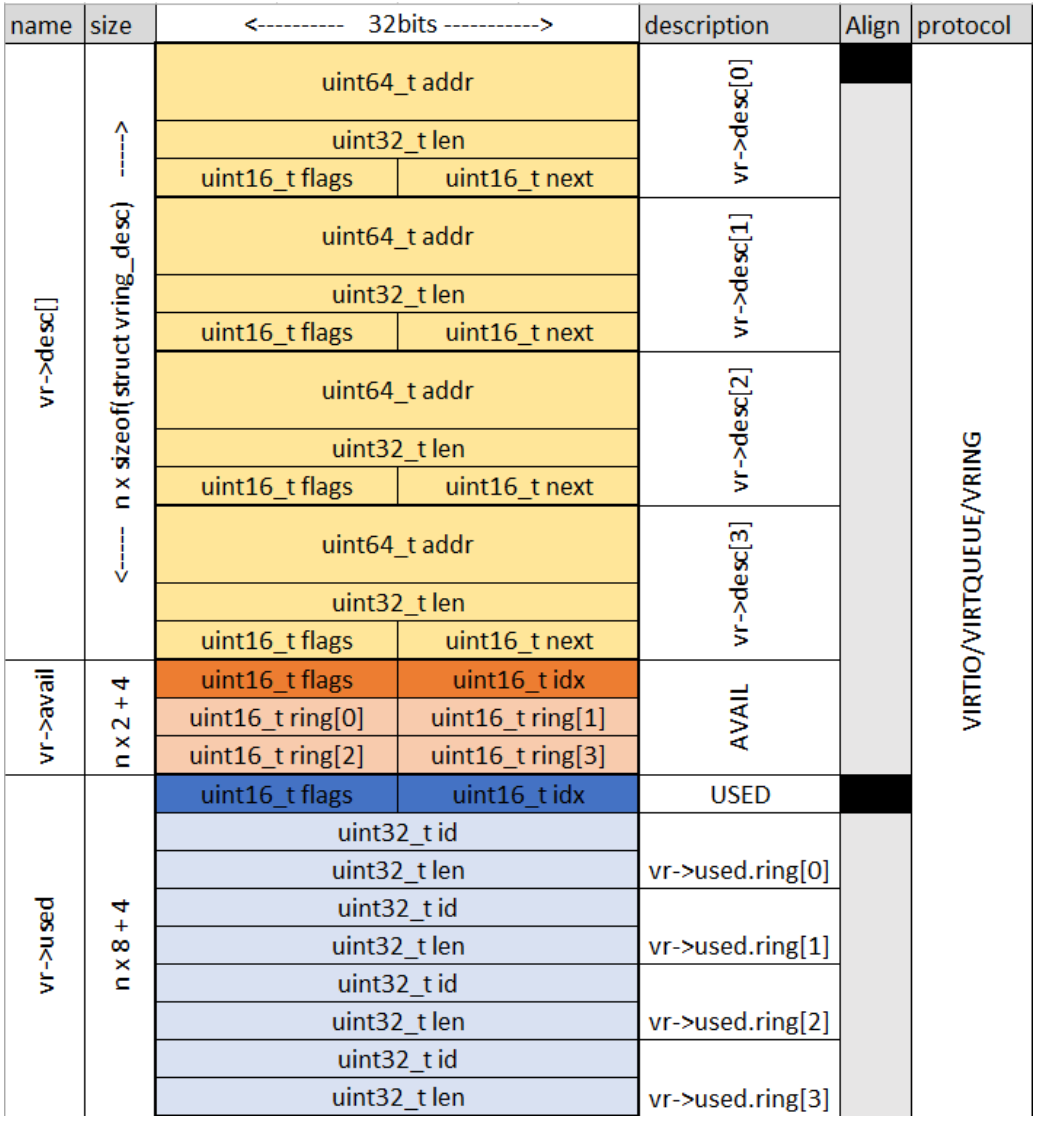
- Physical layer
  - Memory (base, size)
  - Inter-core interrupts (ID)
- MAC layer
  - virtqueue (some info in Linux Kernel doc already)
- Transport layer
  - RMPSG
- Protocol
  - Startup
    - for master/slave(remote) side
  - Receive)
    - for master/slave(remote) side
  - Transmit
    - for master/slave(remote) side

Physical Layer

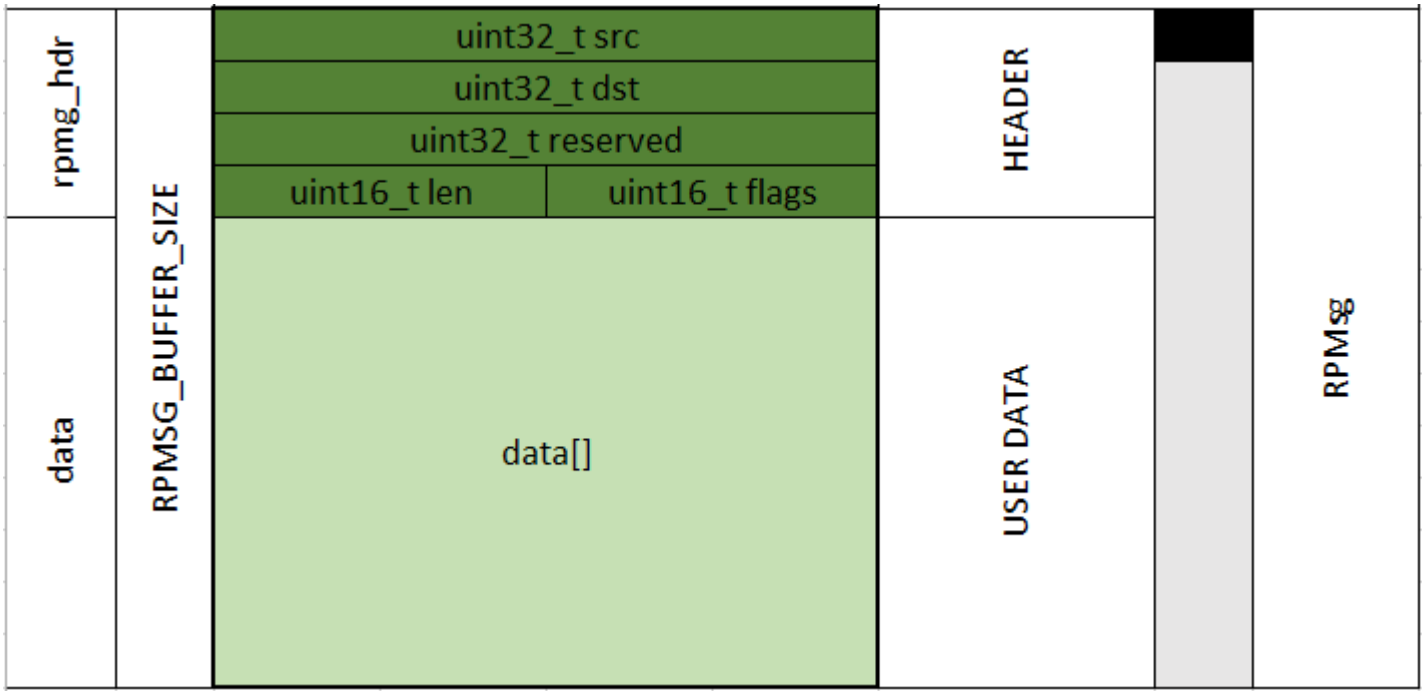
```
92  struct hil_proc proc_table []=
93  {
94      /* CPU node for remote context */
95      {
96          /* CPU ID of master */
97          MASTER_CPU_ID,
98
99          /* Shared memory info - Last field is not used currently */
100         {
101             (void*)SHM_ADDR, SHM_SIZE, 0x00
102         },
103
104         /* VirtIO device info */        /*struct proc_vdev*/
105         {
106             2, (1<<VIRTIO_RPMSG_F_NS), 0,        /*num_vring, dfeatures, gfeatures*/
107
108             /* Vring info */                /*struct proc_vring*/
109             {
110                 /*[0]*/
111                 { /* TX */
112                     NULL, (void*)VRING0_BASE/*phy_addr*/, SHM_SIZE/RPMSG_BUFFER_SIZE/2/*num_descs*/, VRING_ALIGN/*align*/,
113                     /*struct virtqueue, phys_addr, num_descs, align*/
114                     {
115                         /*struct proc_intr*/
116                         VRING0_IPI_VECT,0,0,NULL
117                     }
118                 },
119                 /*[1]*/
120                 { /* RX */
121                     NULL, (void*)VRING1_BASE, SHM_SIZE/RPMSG_BUFFER_SIZE/2, VRING_ALIGN,
122                     {
123                         VRING1_IPI_VECT,0,0,NULL
124                     }
125                 }
126             }
127         },
128
129         /* Number of RPMSG channels */
130         1,  /*num_chnls*/
131
132         /* RPMSG channel info - Only channel name is expected currently */
133         {
134             {"rpmsg-openamp-demo-channel"}  /*chnl name*/
135         },
136
137         /* HIL platform ops table. */
138         &proc_ops,  /*struct hil_platform_ops*/
139
140         /* Next three fields are for future use only */
141         0,
142         0,
143         NULL
144     },
```

- Defined by plaform_info.c
- Not shared between cores, but hardcoded - room for future protocol updates.

MAC Layer – virtqueue memory layout

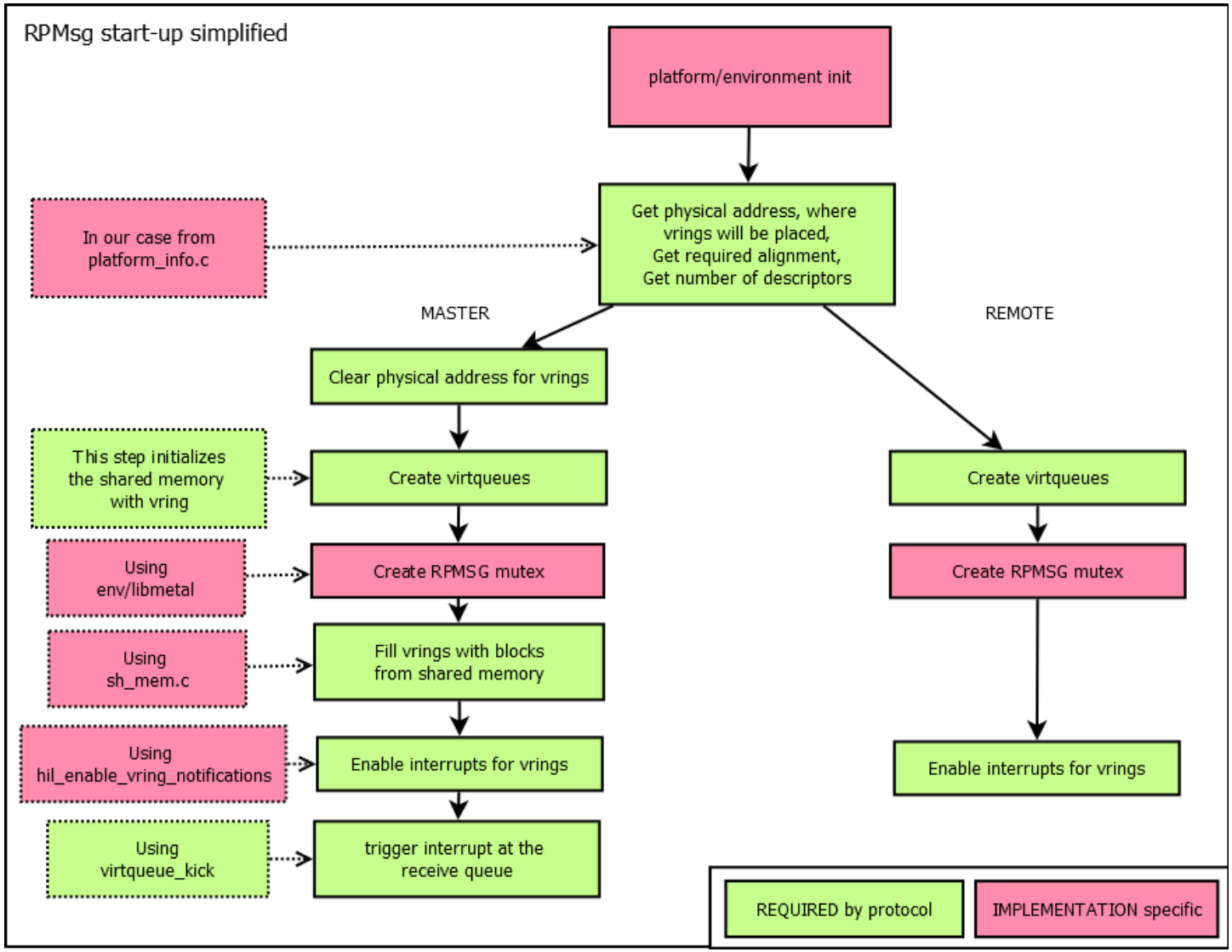


Transport Layer – RPMSG memory layout

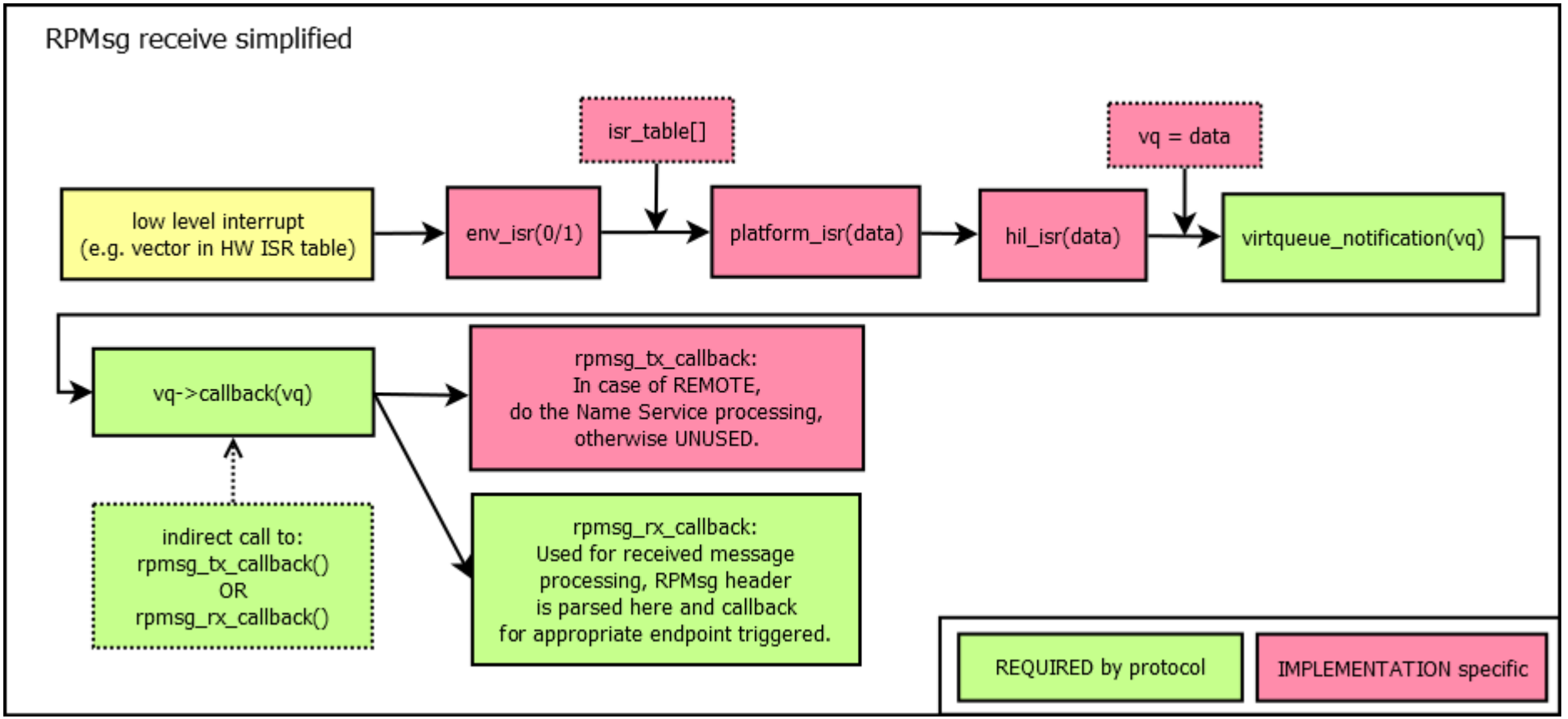


- Flags field not defined completely
- Reserved field could be used for protocol version check when sending a message across the shared memory.

Core Protocol – Startup of RPMsg

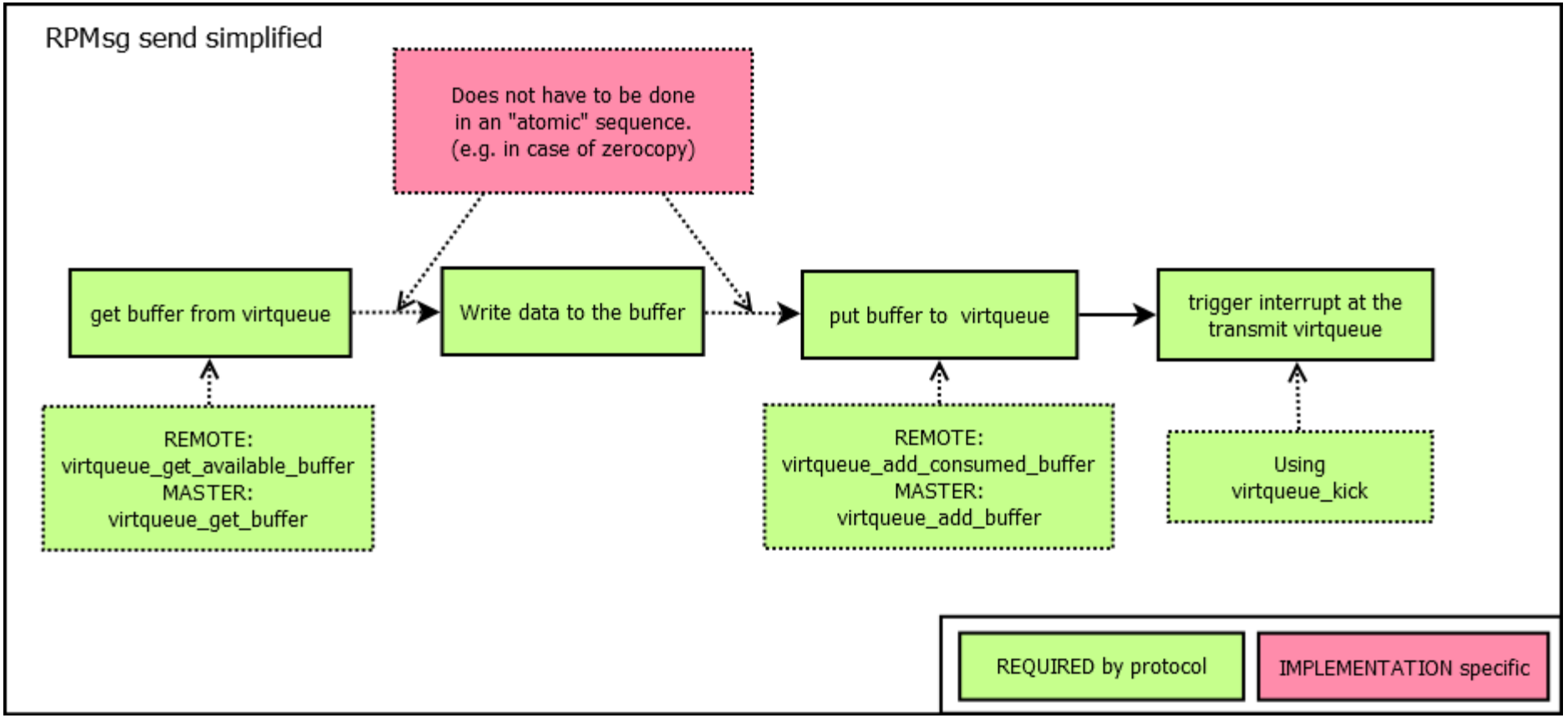


Core Protocol - Receive



- Shouldn’t we reduce the call stack and complexity?

Core Protocol - Transmit



- Simple enough – but current OpenAMP API quite redundant. Do we need: *rpmsg_send_offchannel_raw, rpmsg_trysend_offchannel, rpmsg_trysendto, rpmsg_trysend, rpmsg_sendto*? What about *rpmsg_send* with parameters?
- Send should occur on *endpoint*, not on *channel*.
- Channel may not be part of the *minimum implementation*.

Discussion points / issues

- RPMSG protocol does not have version INFO
- There is no API to check other side is UP
- NameService – should it be required for minimal implementation?
- RPMSG Channel vs Endpoint
  - channel information is not included in protocol header. Should be channel mandatory?
- Future extensions
  - Very large messages support
- Minimalistic implementation using current lower level layers is available at:

  https://github.com/MichalPrincNXP/open-amp/blob/rpmsg_lite/lib/rpmsg_lite/rpmsg_lite.c
  https://github.com/MichalPrincNXP/open-amp/blob/rpmsg_lite/lib/include/openamp/rpmsg_lite.h

  - Used as a *proof of concept*

SECURE CONNECTIONS
FOR A SMARTER WORLD