ALL PROGRAMMABLE
ANY MEDIA
ANY MACHINE
5G
4K/8K
ANY STANDARD
ANY NETWORK
5G Wireless • SDN/NFV • Video/Vision • ADAS • Industrial IoT • Cloud Computing

OpenAMP Remoteproc/RPMsg Decoupling

Purpose of Remoteproc/RPMsg decoupling

- **Use remoteproc only for life cycle management**
  - Platforms:
    - Linux userspace, RTOS, baremetal
  - Features:
    - Specify which firmware to load
    - Allow to boot a firmware without resource table
    - Allow user to register their own “how to get firmware” implementation
    - Enable error detection capability ?
    - Allow user to register their own error handler
    - Allow loading firmware separate from booting the remote
- **Allow user to start RPMsg communication separately to the remoteproc life cycle management**
  - Platforms:
    - Linux userspace, RTOS, baremetal
  - Features:
    - Start RPMsg after remoteproc boots
    - One RPMsg end can detect if the other end has been reset.

2016.4 flow VS new flow without life cycle management

RPMsg slave 2016.4 Flow without life cycle management

- hproc = hil_proc_create(priv_data)
- hil_proc_set_ipi_channel(hproc, channel_name)
- rproc = remoteproc_resource_init(hproc, &rsc, rpmsg_role, rpmsg_cbs)
- Communication start:
  - rpmsg_send(rpmsg_channel, data, len)
- remoteproc_resource_deinit(rproc)

RPMsg slave New flow

- rproc = remoteproc_init(&rsc, vdev_role, priv_data)
- ret= rpmsg_init(rproc, channel_name, rpmsg_role, &rpmsg_dev, rpmsg_create_cb, rpmsg_delete_cb)
- rpmsg_ept = rpmsg_create_ept(rpmsg_dev, rpmsg_dev, ept_addr)
- Communication start:
  - rpmsg_send(rpmsg_ept, data, len)
- rpmsg_deinit(rpmsg_dev);
  - The rpmsg_delete_cb will be called. And in the deinit, it will destroy all the endpoints linked to the rpmsg_dev, after the rpmsg_delete_cb returns.
- remoteproc_deinit(rproc);

2016.4 flow VS new flow without life cycle management

RPMsg master 2016.4 Flow without life cycle management

- hproc = hil_proc_create(priv_data)
- Hil_proc_set_ipi(hproc, vring_id, irq, ipi_data);
- Hil_proc_set_shm(hproc, addr, size);
- hil_proc_set_ipi_channel(hproc, channel_name)
- rproc = remoteproc_resource_init(hproc, &rsc, rpmsg_role, rpmsg_cbs)
- Communication start:
  - rpmsg_send(rpmsg_channel, data, len)
- remoteproc_resource_deinit(rproc)

RPMsg master New flow

- rproc = remoteproc_init(&rsc, vdev_role, priv_data)
- remoteproc_set_shm(&rproc, shm_addr, shm_size)
- remoteproc_set_ipi(&rproc, vring_id, irq, ipi_data)
- ret= rpmsg_init(rproc, channel_name, rpmsg_role, &rpmsg_dev, rpmsg_create_cb, rpmsg_delete_cb)
- rpmsg_ept = rpmsg_create_ept(rpmsg_dev, rpmsg_dev, ept_addr)
- Communication start:
  - rpmsg_send(rpmsg_ep, data, len)
- rpmsg_deinit(rpmsg_dev);
  - The rpmsg_delete_cb will be called. And in the deinit, it will destroy all the endpoints linked to the rpmsg_dev, after the rpmsg_delete_cb returns.
- remoteproc_deinit(rproc);

New Flow with life cycle management but without RPMsg

Remoteproc master New flow

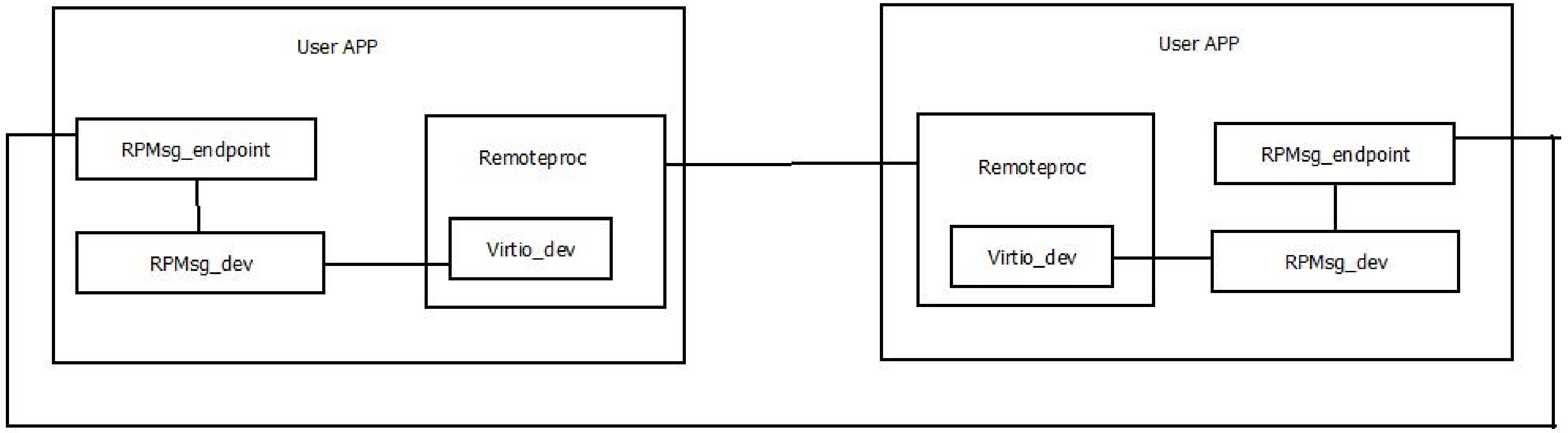
- rproc = remoteproc_init(0, 0, priv_data)
- remoteproc_register_err(err, err_callback); ??
- remoteproc_boot(rproc, fw_name)
- remoteproc_shutdown(rproc)

New Flow with life cycle management and RPMsg

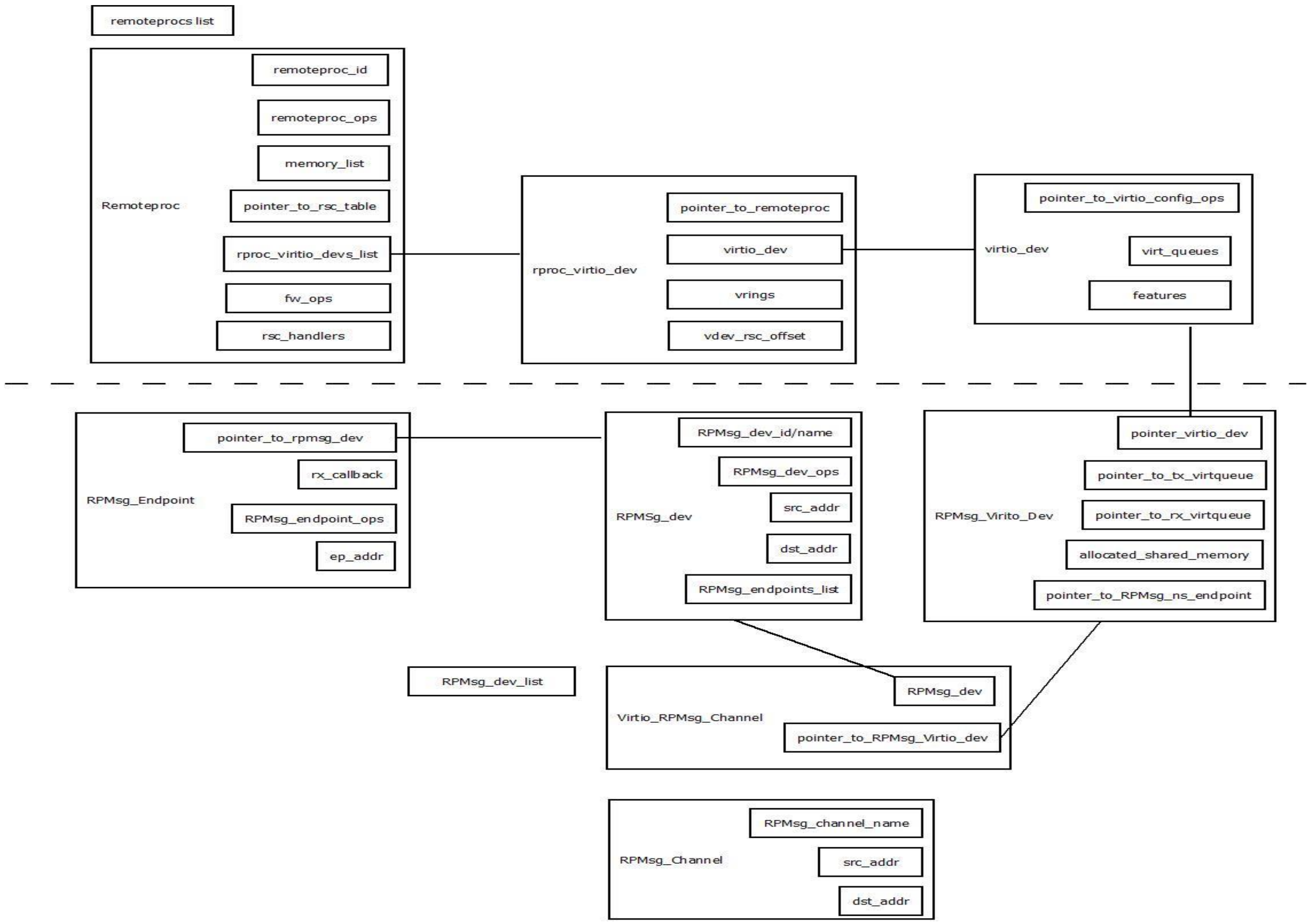
Remoteproc master New flow

- rproc = remoteproc_init(0, 0, priv_data)
- remoteproc_register_err(err, err_callback); ??
- remoteproc_boot(rproc, fw_name) // parse ELF firmware, pass rsc from the firmware, create virtio devices
- remoteproc_set_shm(&rproc, shm_addr, shm_size)
- remoteproc_set_ipi(&rproc, vring_id, irq, ipi_data)
- ret= rpmsg_init(rproc, channel_name, rpmsg_role, &rpmsg_dev, rpmsg_create_cb, rpmsg_delete_cb)
- rpmsg_ept = rpmsg_create_ept(rpmsg_dev, rpmsg_dev, ept_addr)
- Communication start:
  – rpmsg_send(rpmsg_ept, data, len)
- rpmsg_deinit(rpmsg_dev);
  – The rpmsg_delete_cb will be called. And in the deinit, it will destroy all the endpoints linked to the rpmsg_dev, after the rpmsg_delete_cb returns.
- remoteproc_shutdown(rproc)
- remoteproc_deinit(rproc);

Components Architecture Overview



Connections between components



OpenAMP in Linux userspace

- OpenAMP is a library, user can use it in different systems such as Linux, FreeRTOS and Baremetal
- The APIs to users are the same such as remoteproc_init(), rpmsg_send() and so on.
- The underline implementation can be different
- In Linux userspace,
  - Remoteproc_init() will probe the remoteproc kernel driver
  - Remoteproc_boot() will use remoteproc kernel driver sysfs APIs to set the firmware and boot the remote.
  - remoteproc_shutdown() will use remoteproc kernel driver sysfs APIs to shutdown the remoteproc
  - rpmsg_XXX() operations
    - Default rpmsg operations is rpmsg/virtio in Kernel userspace
    - In future, when rpmsg char dev is available in kernel, will also add support to use the rpmsg char dev.

Questions

- How can remoteproc master know the status of the slave
  - Sofheart beat?
    - Introduce a new resource type in the resource table:

```
Struct sw_heartbeat {
      u32 challenge; // master to update
      u32 response; // slave to set inverting the “challenge” once it is notified
      u32 nsec_expire; // heartbeat interval
 };
```

- What to do with the error detection?
  - What errors to detect?
    - Crash and what else?
  - What to do if there is an error happens?
    - Just call the error handler if it is registered?
    - Shutdown the remote after the error hander is called?
- in case of one RPMsg end got restarted, how to restart the communication
  - Use the virtio_dev status field ?
    - Whenever one ends starts RPMsg, it toggle one bit in the vdev status field.
    - Will need to notify the other end when virtio_dev status is set
    - When the other end get the notification, it will check the virtio dev status, if it is changed, it will restart the communication.

Linux Questions

- How to boot with Linux without resource table in the firmware
  - Remoteproc kernel patch is required: Support empty resource table: [PATCH v2 19/19] remoteproc: core: Support empty resource tables
- How to use Linux remoteproc sysfs APIs to boot/shutdown remote
  - Remoteproc kernel patch is required: [PATCH v3 0/2] remoteproc: Add sysfs interface
- How to register platform devices from Linux remoteproc kernel driver?
  - This is in order for userspace application to user the device as UIO/VFIO with libmetal
  - How to know what devices to use for OpenAMP application?
    - Add a resource table type e.g. RSC_DEV for this devices?
- How to authenticate firmware with Linux remoteproc kernel driver?