Linux RPMsg framework overview

This article gives information about the Linux® Remote Processor Messaging (RPMsg) framework. The RPMsg framework is a virtio-based messaging bus that allows a local processor to communicate with remote processors available on the system.

Contents

1 Framework purpose

The Linux® RPMsg framework is a messaging mechanism implemented on top of the virtio framework[1][2] to communicate with a remote processor. It is based on virtio vrings to send/receive messages to/from the remote CPU over shared memory.

The vrings are uni-directional, one vring is dedicated to messages sent to the remote processor, and the other vring is used for messages received from the remote processor. In addition, shared buffers are created in memory space visible to both processors.

The Mailbox framework is then used to notify cores when new messages are waiting in the shared buffers.

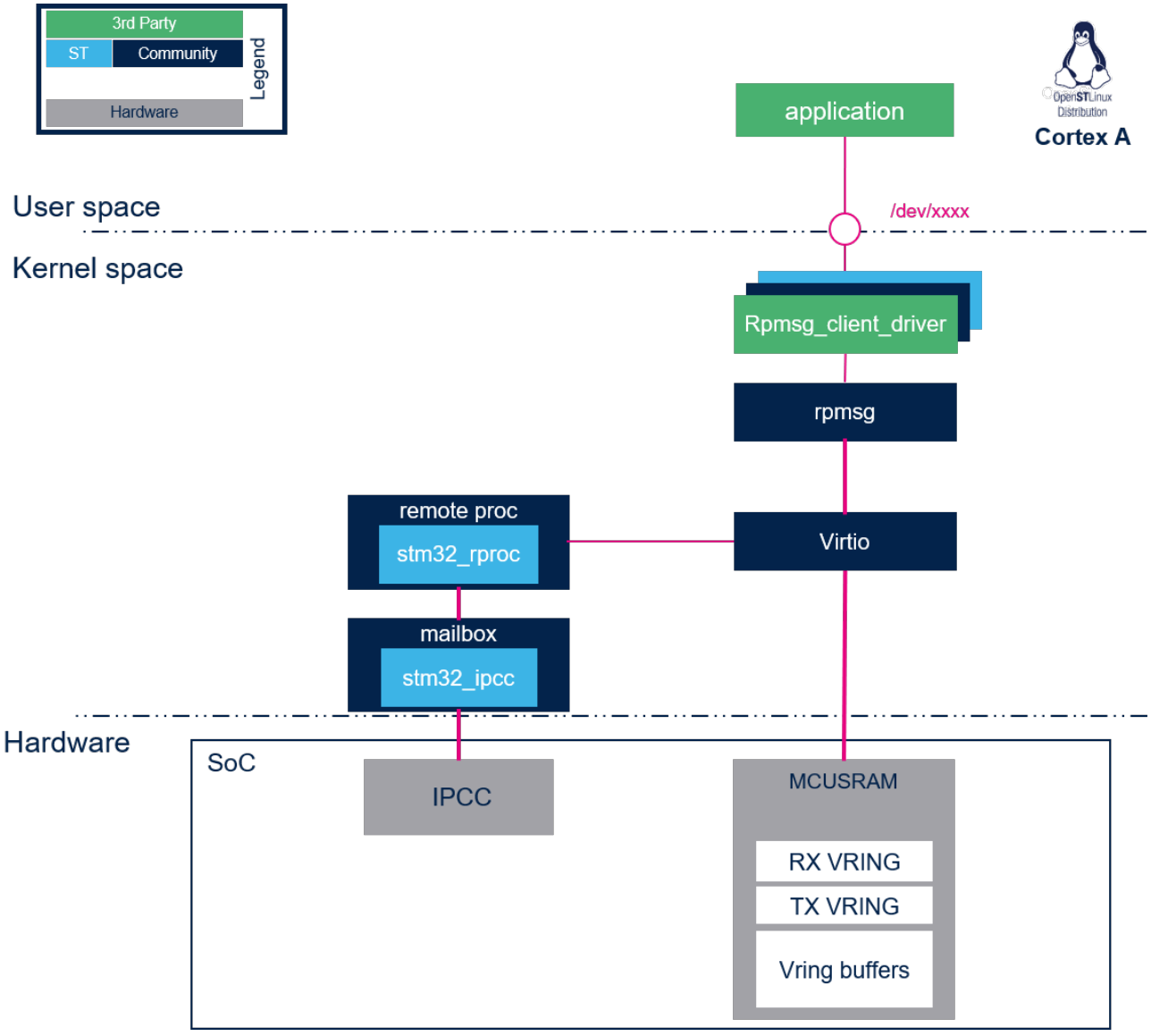
Relying on these frameworks, The RPMsg framework implements a communication based on channels. The channels are identified by a textual name and have a local (“source”) RPMsg address, and a remote (“destination”) RPMsg address”.

On the remote processor side, a RPMSG framework must also be implemented. Several solutions exist, we recommend using OpenAMP

Overviews of the communication mechanisms involved are available at:

- OpenAMP wiki [3][4]

2 System overview



2.1 Component description

- **remoteproc**: The remoteproc framework allows different platforms/architectures to control (power on, load firmware, power off) the remote processors. This framework also adds rpmsg virtio devices for remote processors that support the RPMsg protocol. More details on this framework are available in the remote proc framework page.
- **virtio**: VirtIO framework that supports virtualization. It provides an efficient transport layer based on a shared ring buffer (vring). For more details about this framework please refer to the link below:
  - Virtio: An I/O virtualization framework for Linux [1]
  - virtio introduction - SlideShare [2]
- **rpmsg**: A virtio-based messaging bus that allows kernel drivers to communicate with remote processors available on the system. It provides the messaging infrastructure, facilitating the writing of wire-protocol messages by client drivers. Client drivers can then, in turn, expose appropriate user space interfaces if needed.
- **rpmsg_client_driver** is the client driver that implements a service associated to the remote processor. This driver is probed by the RPMsg framework when an associated service is requested by a remote processor using a "new service announcement" RPMsg message.

2.2 RPMsg definitions

To implement an Rpmsg client, **channel** and **endpoint** concepts need to be understood for a good understanding of the framework.

- RPMsg channel:

An RPMsg client is associated to a communication channel between master and remote processors. This RPMsg client is identified by the textual **service name**, registered in the RPMsg framework. The communication channel is established when a match is found between the local service name registered and the remote service announced.

- RPMsg endpoint:

The RPMsg endpoints provide logical connections through an RPMsg channel. An RPMsg endpoint has a unique source address and associated call back function, allowing the user to bind multiple endpoints on the same channel. When a client driver creates an endpoint with the local address, all the inbound messages with a destination address equal to the endpoint local address are routed to that endpoint. Notice that every channel has a default endpoint, which enables applications to communicate without even creating new endpoints.

2.3 API description

The User API usage is described in Linux kernel RPMsg documentation [5]

3 Configuration

3.1 Kernel configuration

The RPMsg framework is automatically enabled when the remote STM32_RPROC configuration is activated.

3.2 Device tree configuration

No device tree configuration is needed. The Memory region allocated for rpmsg buffers is declared in the remote proc framework device tree.

4 How to use the framework

The Rpmsg framework is used by linux driver client. For details and an example of a simple client, please refer to associated Linux documentation [5]

5 How to trace and debug the framework

5.1 How to trace

RPMsg and virtio dynamic debug traces can be added using the following commands:

```
echo -n 'file virtio_rpmsg_bus.c +p' > /sys/kernel/debug/dynamic_debug/control
echo -n 'file virtio_ring.c +p' > /sys/kernel/debug/dynamic_debug/control
```

6 References

- [1.01.1] An I/O virtualization framework for Linux
- [2.02.1] virtio introduction - SlideShare
- RPMsg Messaging Protocol
- RPMsg Communication Flow
- [5.05.1] Linux kernel rpmsg documentation