OpenAMP: Out-Of-Band Transmission of Large Buffers

Authors: Ed Mooring (Xilinx/Linaro), Ben Levinsky (Xilinx)

Linaro connect
San Diego 2019

Motivations

- Performance
  - Small fixed-size buffers do not work well with large amounts of data (e.g. video, or radar streams)
  - Multiple copy operations per message
  - Take advantage of DMA-capable devices
- Community Acceptance

Requirements

- **Zero-Copy of Payload**
- **Arbitrarily-large Buffers**
- **Adaptable to Constrained Memory Environments**
- **Sender-Managed Resources**
- **Compatible with existing implementation**

Architecture

- Buffer Pool
  - Chunk of contiguous physical memory shared between the two processors
  - Distinct from rpmsg/vring memory
  - Divided into fixed size buffers
- Buffer Management
- Per-Connection Data
  - One buffer pool per connection
  - One way flow of large messages
  - One RPMsg endpoint for each processor

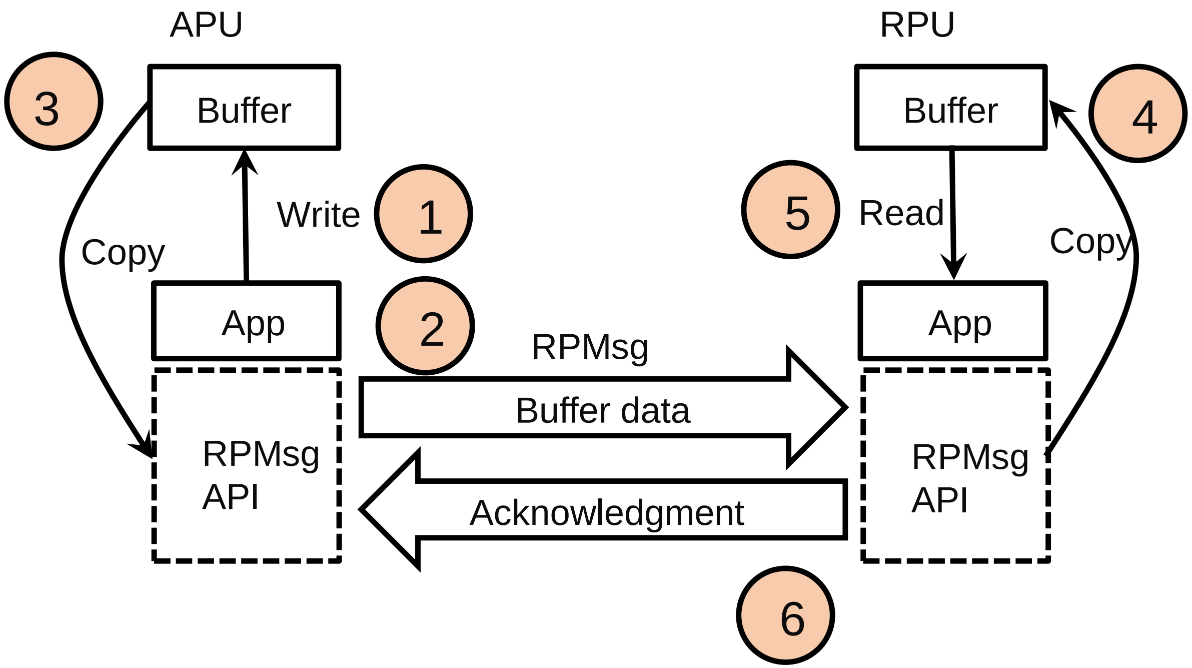
Protocol

- Rides “atop” the existing RPMsg protocol
- Takes the unstructured RPMsg payload and adds a new header structure:
  - OOB Flag/Type (32 bits)
  - Type-dependent additional fields (32 bits each)
- Types
  - Init – Informs receiver of the start of the connection and the resources available
  - Data – Index of a buffer and the length of the data
  - Acknowledgment – From the receiver, indicates it has finished processing the buffer
  - In-band data – The rest of the message is unstructured data (can be sent in either direction)

API

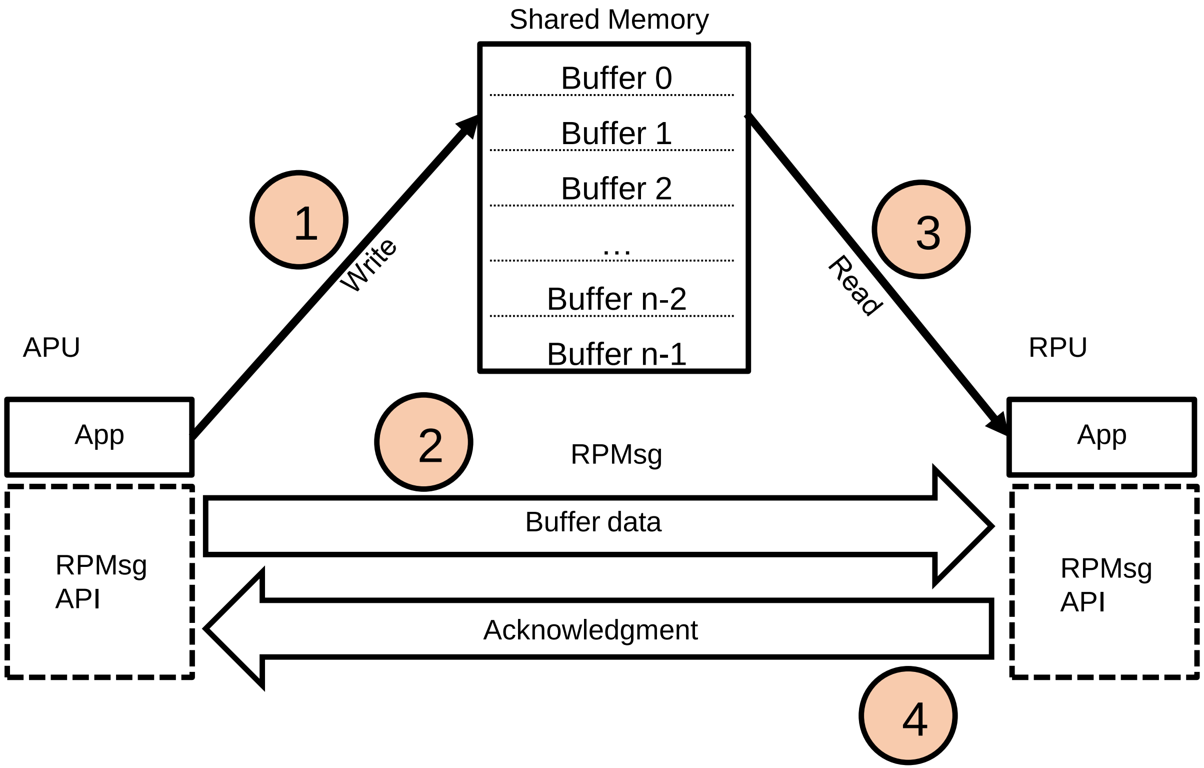
- Buffers are passed as an index, rather than an address. The two endpoints might have different mappings for the same physical memory.
- Buffer Management
  - Initialize the buffer pool
  - Get a buffer
  - Put a buffer
- Message Transmission
  - Send a buffer
  - Send an in-band data message
  - Acknowledge a buffer

RPMsg Without Zero Copy



1. APU application writes to buffer.
2. APU application sends buffer via RPMsg.
3. APU RPMsg library copies buffer into message.
4. RPU RPMsg library copies message into buffer.
5. RPU application reads buffer.
6. RPU application sends acknowledgment via RPMsg.

Out of Band Zero Copy using RPMsg



1. APU application writes to buffer.
2. APU application sends buffer index via RPMsg.
3. RPU application reads buffer at index.
4. RPU application sends acknowledgment via RPMsg.

Future Work

- Provide reset/crash recovery in the protocol
- Provide API calls to reset the connection
- Investigate a higher-level API

Proof of Concept Implementation

- Cortex-R5 serves Acks when successfully receiving a RPMsg Out of Band message
- Cortex-A53 sends RPMsg packets with Out of Band message encoded

Cortex-R5 Server waiting for RPMsg endpoint to send buffers

- Previously buffer has only been application-specific data
- Add support to mark packets with new structure where instead of application data is present in the packet, the information is an offset into shared memory space that is read/write accessible to both RPU and APU
- Send Ack once message is received

Out Of Band Packet Structure
Packet Type
Shared Buffer Index
Packet Length

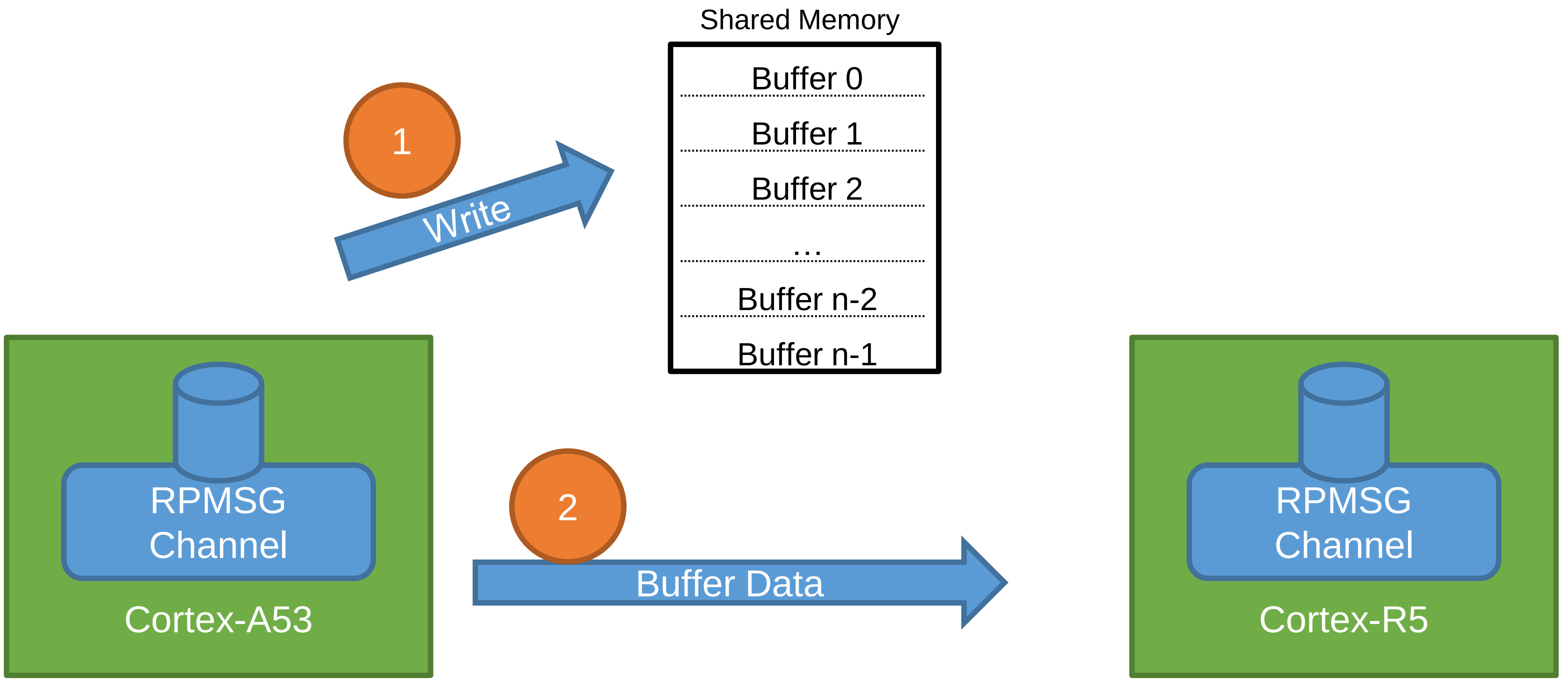
Cortex-A53 sending Packets

- Message contents are written directly to shared memory space instead of copy to buffers
- RPMsg buffer content is offset into shared memory space
- Construct packet with type OUT OF BAND

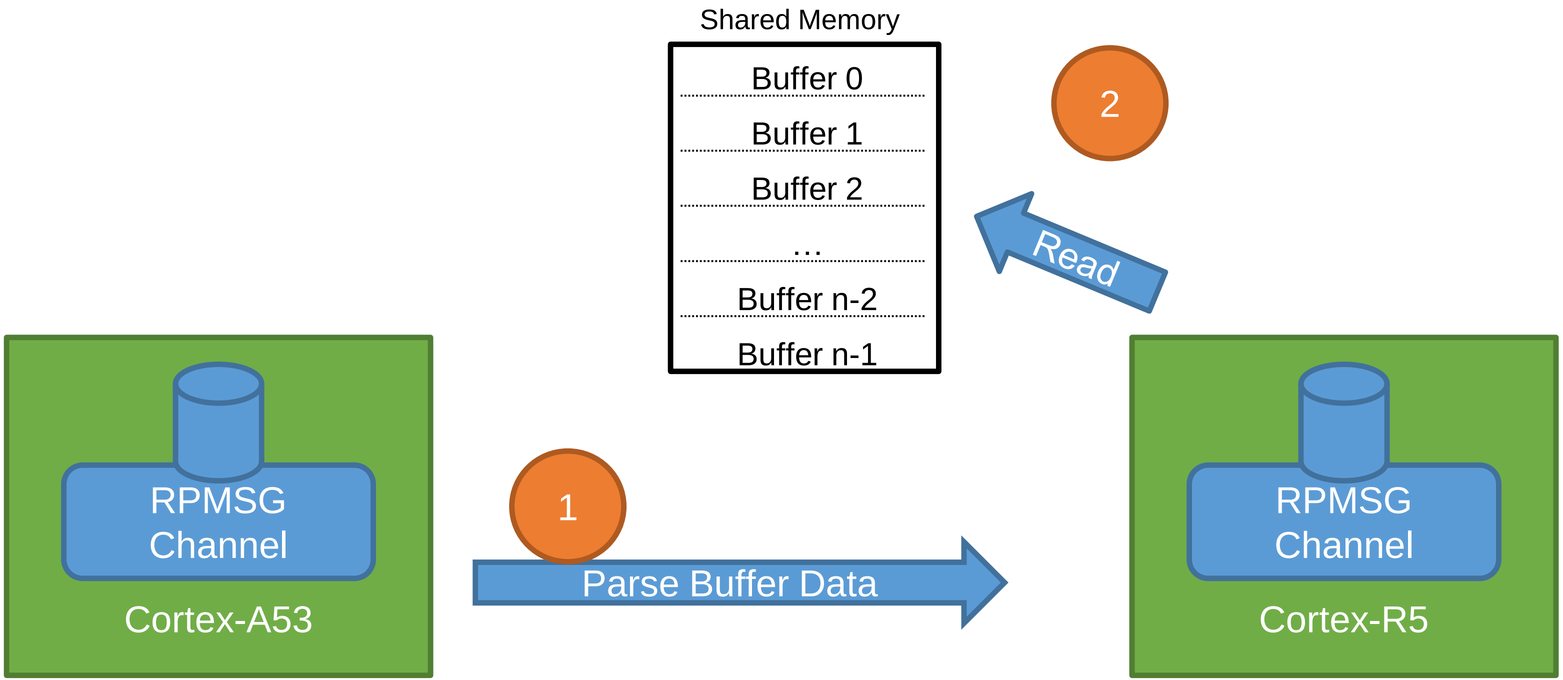
RPMsg Memory Layout	
Source Address (32 bit)	
Destination Address (32 bit)	
Reserved (32 bit)	
Length (16 bit)	Flags (16 bit)
User Payload	

Out of Band RPMsg Memory Layout		
Source Address (32 bit)		
Destination Address (32 bit)		
Reserved (32 bit)		
Length (16 bit)	Flags (16 bit)	
Type (32 bit)	Buf Idx (32 bit)	Packet Len (32 bit)

Cortex A-53 Sending Packets to Cortex R-5



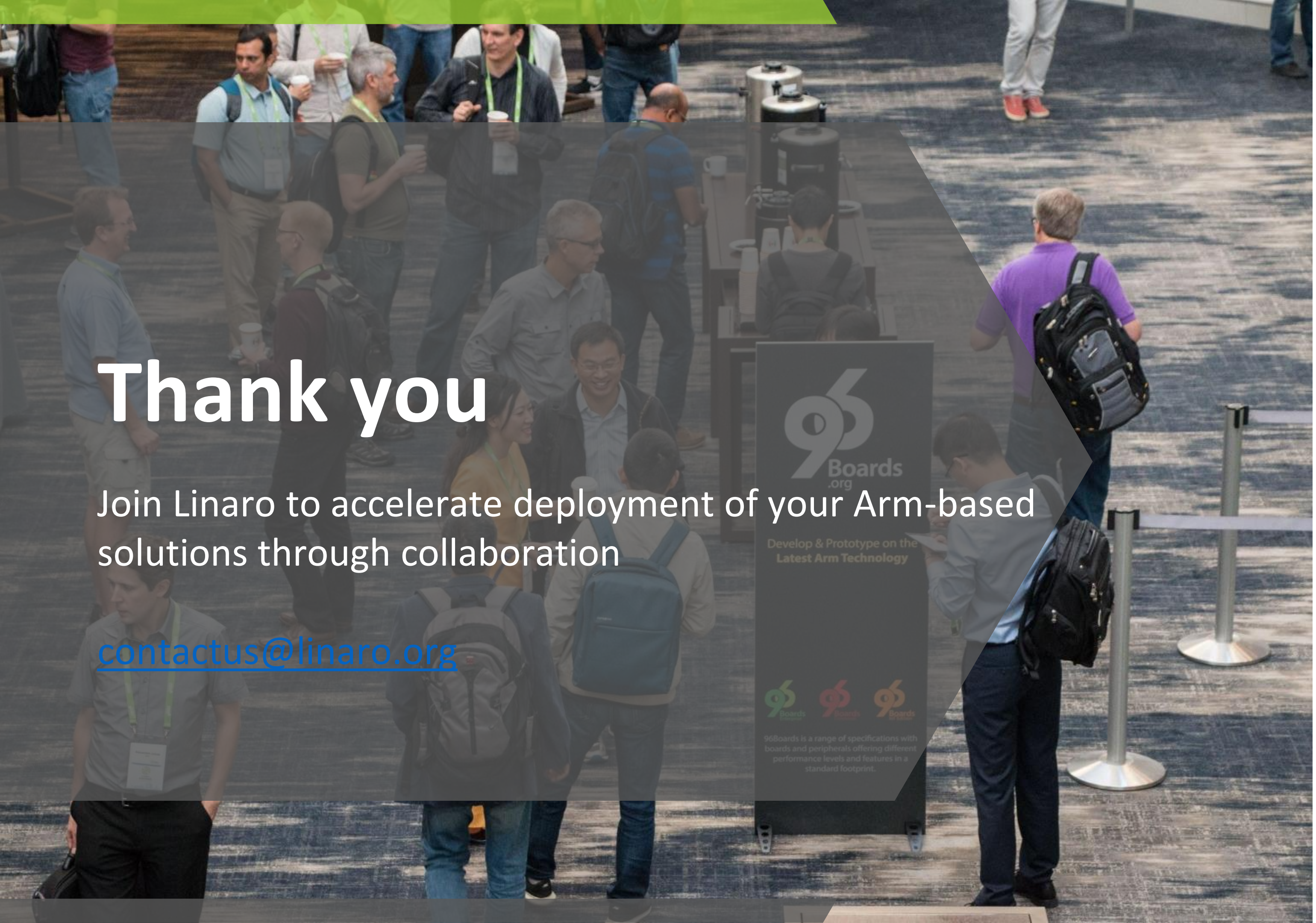
Cortex R-5 Reading Packets



Thank you

Join Linaro to accelerate deployment of your Arm-based solutions through collaboration

contactus@linaro.org



Linaro connect San Diego 2019