

# Using Virtio To talk with Remote Cores

Ohad Ben-Cohen <[ohad@wizery.com](mailto:ohad@wizery.com)>





What ?

Why ??

1. simple (but allow both kernel and userland to talk with remote services)
2. performance
3. generic
4. upstream
5. focus on Linux, but
6. wire protocol is BSD

show me

the code

```

static void sample_rx_callback(struct rpmsg_channel *rpdev,
                               void *data, int len, void *priv, u32 src)
{
    /* profit ! */
}

static int sample_probe(struct rpmsg_channel *rpdev)
{
    return rpmsg_send(rpdev, "dude", 4);
}

static struct rpmsg_device_id sample_id_table[] = {
    { .name = "rpmsg-client-sample" }, { },
};

static struct rpmsg_driver sample_client = {
    ...
    .id_table    = sample_id_table,
    .probe       = sample_probe,
    .callback    = sample_rx_callback,
};

static int __init client_sample_init(void)
{
    return register_rpmsg_driver(&sample_client);
}

```

```
static void sample_rx_callback(struct rpmsg_channel *rpdev,
                              void *data, int len, void *priv, u32 src)
{
    /* profit ! */
}

static int sample_probe(struct rpmsg_channel *rpdev)
{
    return rpmsg_send(rpdev, "dude", 4);
}
```

**1**

```
static struct rpmsg_device_id sample_id_table[] = {
    { .name = "rpmsg-client-sample" }, { },
};
```

```
static struct rpmsg_driver sample_client = {
    ...
    .id_table    = sample_id_table,
    .probe       = sample_probe,
    .callback    = sample_rx_callback,
};
```

```
static int __init client_sample_init(void)
{
    return register_rpmsg_driver(&sample_client);
}
```

```
static void sample_rx_callback(struct rpmsg_channel *rpdev,
                              void *data, int len, void *priv, u32 src)
{
    /* profit ! */
}

static int sample_probe(struct rpmsg_channel *rpdev)
{
    return rpmsg_send(rpdev, "dude", 4);
}

static struct rpmsg_device_id sample_id_table[] = {
    { .name = "rpmsg-client-sample" }, { },
};
```

2

```
static struct rpmsg_driver sample_client = {
    ...
    .id_table    = sample_id_table,
    .probe       = sample_probe,
    .callback    = sample_rx_callback,
};

static int __init client_sample_init(void)
{
    return register_rpmsg_driver(&sample_client);
}
```



```

static void sample_rx_callback(struct rpmsg_channel *rpdev,
                              void *data, int len, void *priv, u32 src)
{
    /* profit ! */
}

static int sample_probe(struct rpmsg_channel *rpdev)
{
    return rpmsg_send(rpdev, "dude", 4);
}

static struct rpmsg_device_id sample_id_table[] = {
    { .name = "rpmsg-client-sample" }, { },
};

static struct rpmsg_driver sample_client = {
    ...
    .id_table    = sample_id_table,
    .probe       = sample_probe,
    .callback    = sample_rx_callback,
};

```

3

```

static int __init client_sample_init(void)
{
    return register_rpmsg_driver(&sample_client);
}

```

```
static void sample_rx_callback(struct rpmsg_channel *rpdev,  
                              void *data, int len, void *priv, u32 src)  
{  
    /* profit ! */  
}
```

4

```
static int sample_probe(struct rpmsg_channel *rpdev)  
{  
    return rpmsg_send(rpdev, "dude", 4);  
}
```

```
static struct rpmsg_device_id sample_id_table[] = {  
    { .name = "rpmsg-client-sample" }, { },  
};
```

```
static struct rpmsg_driver sample_client = {  
    ...  
    .id_table    = sample_id_table,  
    .probe       = sample_probe,  
    .callback    = sample_rx_callback,  
};
```

```
static int __init client_sample_init(void)  
{  
    return register_rpmsg_driver(&sample_client);  
}
```

# 5

```
static void sample_rx_callback(struct rpmsg_channel *rpdev,  
                               void *data, int len, void *priv, u32 src)
```

```
{  
    /* profit ! */  
}
```

```
static int sample_probe(struct rpmsg_channel *rpdev)
```

```
{  
    return rpmsg_send(rpdev, "dude", 4);  
}
```

```
static struct rpmsg_device_id sample_id_table[] = {
```

```
    { .name = "rpmsg-client-sample" }, { },  
};
```

```
static struct rpmsg_driver sample_client = {
```

```
    ...  
    .id_table    = sample_id_table,  
    .probe      = sample_probe,  
    .callback    = sample_rx_callback,  
};
```

```
static int __init client_sample_init(void)
```

```
{  
    return register_rpmsg_driver(&sample_client);  
}
```



**status**

## Messages in this thread

- *First message in thread*
- **Linus Torvalds**
- *Stephen Rothwell*
- *Steven Rostedt*
- *Junio C Hamano*
- *Geert Uytterhoeven*
- *Andrew Morton*
- *Ohad Ben-Cohen*
- *Grant Likely*
- *Arnd Bergmann*
- *Rusty Russell*
- *Carlos Chinae*
- *Linus Walleij*
- *Carlos Chinae*
- *Artem Bityutskiy*

(Ad)

   Web  lkml.org

**From** Linus Torvalds <>  
**Date** Thu, 19 Jan 2012 15:58:28 -0800    
**Subject** Linux 3.3-1 out - merge window closed

So the subject says it all. It's been two weeks(+a day), and 3.3-rc1 is now out.

There are a couple of trees I haven't merged on purpose, and there may be a few trees I overlooked by mistake. The "on purpose" ones were things that looked unfamiliar and I felt I didn't have the bandwidth to check. The "mistake" ones would just be things I missed due to being busy.

And it really was a pretty busy merge window. I don't know *\*why\** it felt so busy, though. In pure numbers, the merge window seems to have been pretty normal - the number of merges and regular commits are right where you'd expect them. Part of it was spending what felt like (and I think was) a couple of days chasing down two independent suspend/resume regressions on my laptop, part of it was a couple of just bad pull requests, and some of it was some of the independent discussions that were on-going. But none of that is unheard of, so what do I know..

Anyway, it's out now, and I'm taking off early for a weekend of beer, skiing and poker (not necessarily in that order: "don't drink and ski"). No email.

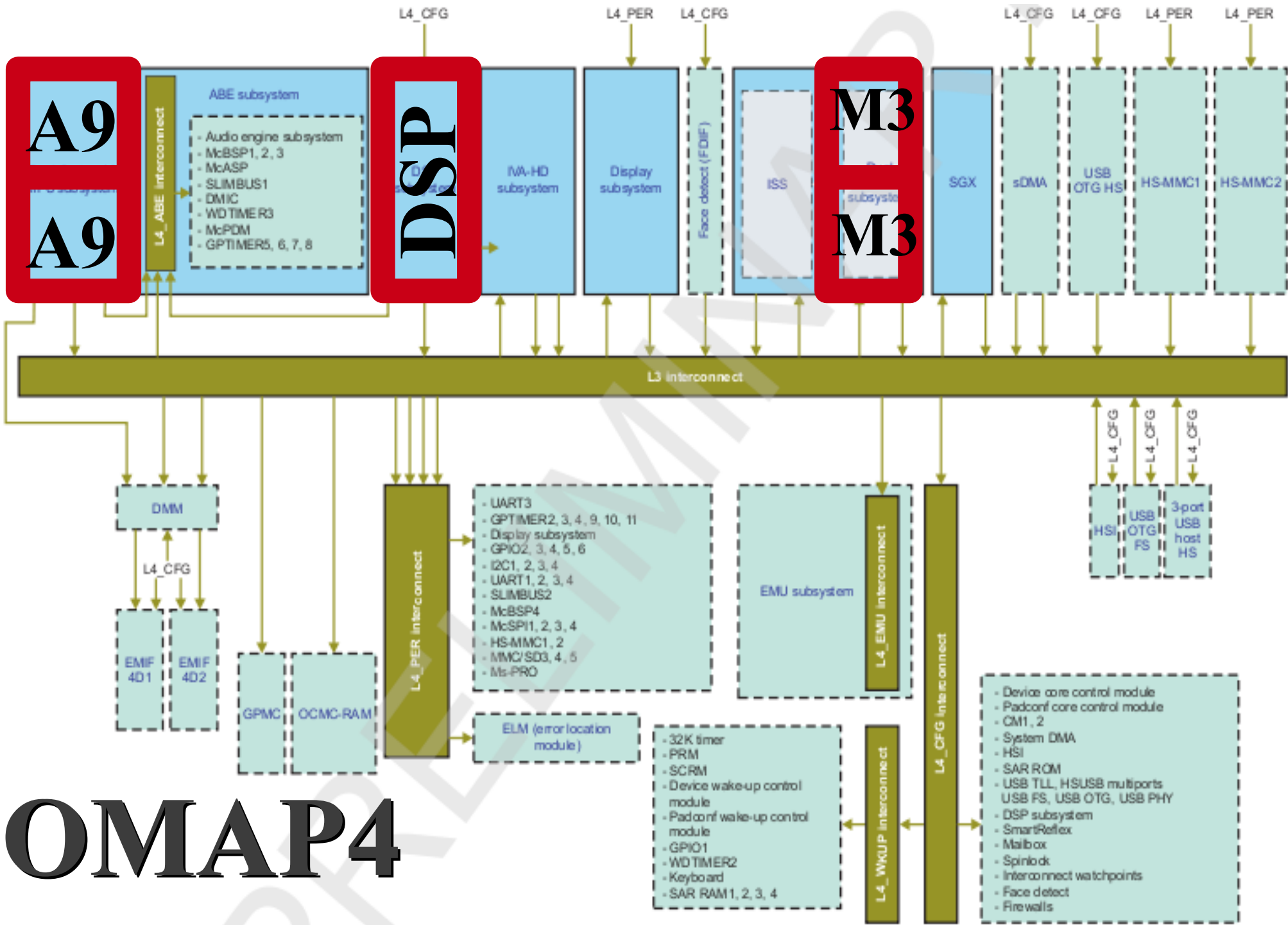
So if you felt that your pull request was overlooked by mistake (or intentionally, but really not so scary that you think I should have a really easy time checking it), you have a couple of days to marshal your arguments for why I should pull it after all.

And if you didn't send your pull request in time: "Phhhthrthtpt!". No arguments for that one.

details

# small letters

- **On-chip processors**
- **No bus** (RapidIO ? PCIe ?)
- **Can access memory**
- **Can kick each other**
- **... common SoC layout**



# OMAP4



# 1. Control


drivers/amp/remoteproc

- **Load a firmware**
- **Allocate resources**
- **Boot**

# rproc\_...()

- `int rproc_boot(struct rproc *)`
- `int rproc_shutdown(struct rproc *)`
- `rproc *rproc_get_by_name(char *)`
- `int rproc_put(struct rproc *)`

Try not  
to use  
these



## Two different refcounts!

# 2. Input/Output

**drivers/amp/rpmsg**

- **Send messages**
- **Receive messages**
- **Multiple channels**

# rpmsg\_...()

- **int rpmsg\_send(rpdev, data, len)**
- **int rpmsg\_sendto(rpdev, data, len, dst)**
- **int rpmsg\_send\_offchannel(rpdev, src, dst, data, len)**
  
- **int rpmsg\_trysend(rpdev, data, len)**
- **int rpmsg\_trysendto(rpdev, data, len, dst)**
- **int rpmsg\_trysend\_offchannel(rpdev, src, dst, data, len)**
  
- **struct rpmsg\_endpoint \***  
  **rpmsg\_create\_ept(rpdev, callback, priv, addr)**
- **void rpmsg\_destroy\_ept(struct rpmsg\_endpoint \*);**

# On channels

- **Rpmsg a bus, channels: devices**
- **Match by name**
- **Carry src + dst addresses**
- **Same physical medium**
- **Can be dynamically allocated**
- **Drivers, callbacks and probe()**
- **Userland ? Sure, but...**

Under the hood:

**VirtIO**

# Virtqueues

- A transport abstraction
- API for posting buffers for consumption

```
struct virtqueue_ops {  
    int (*add_buf)(struct virtqueue *vq,  
                  struct scatterlist sg[],  
                  unsigned int out_num,  
                  unsigned int in_num,  
                  void *data);  
    void (*kick)(struct virtqueue *vq);  
    void *(*get_buf)(struct virtqueue *vq, unsigned int *len);  
    void (*disable_cb)(struct virtqueue *vq);  
    bool (*enable_cb)(struct virtqueue *vq);  
};
```

- Each device has one (or more) virtqueues
- A virtio driver uses the above API for I/O
- Asymmetric in nature: Guest ↔ Host

# Virtqueues

- Was collapsed into a single implementation:

commit 7c5e9ed0c84e7d70d887878574590638d5572659

Author: Michael S. Tsirkin <mst@redhat.com>

Date: Mon Apr 12 16:19:07 2010 +0300

virtio\_ring: remove a level of indirection

We have a single virtqueue\_ops implementation, and it seems unlikely we'll get another one at this point. So let's remove an unnecessary level of indirection: it would be very easy to re-add it if another implementation surfaces.

Signed-off-by: Michael S. Tsirkin <mst@redhat.com>

Signed-off-by: Rusty Russell <rusty@rustcorp.com.au>

drivers/virtio/virtio\_ring.c | 36 ++++++++-----

include/linux/virtio.h | 71 ++++++++-----

2 files changed, 34 insertions(+), 73 deletions(-)



# vring

- The only virtqueue implementation today

```
struct vring
```

```
{
```

```
    // The actual descriptors (16 bytes each)  
    struct vring_desc desc[num];
```

```
    // A ring of available descriptor heads with free-running index.
```

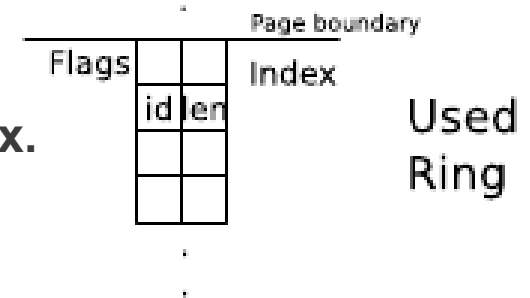
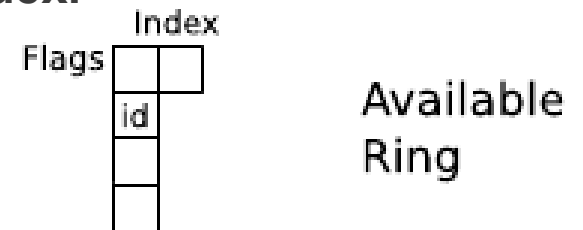
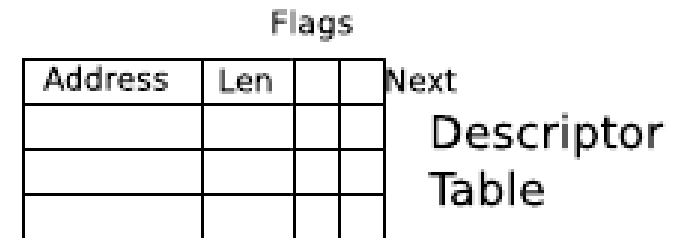
```
    __u16 avail_flags;  
    __u16 avail_idx;  
    __u16 available[num];
```

```
    // Padding to the next align boundary.  
    char pad[];
```

```
    // A ring of used descriptor heads with free-running index.
```

```
    __u16 used_flags;  
    __u16 used_idx;  
    struct vring_used_elem used[num];
```

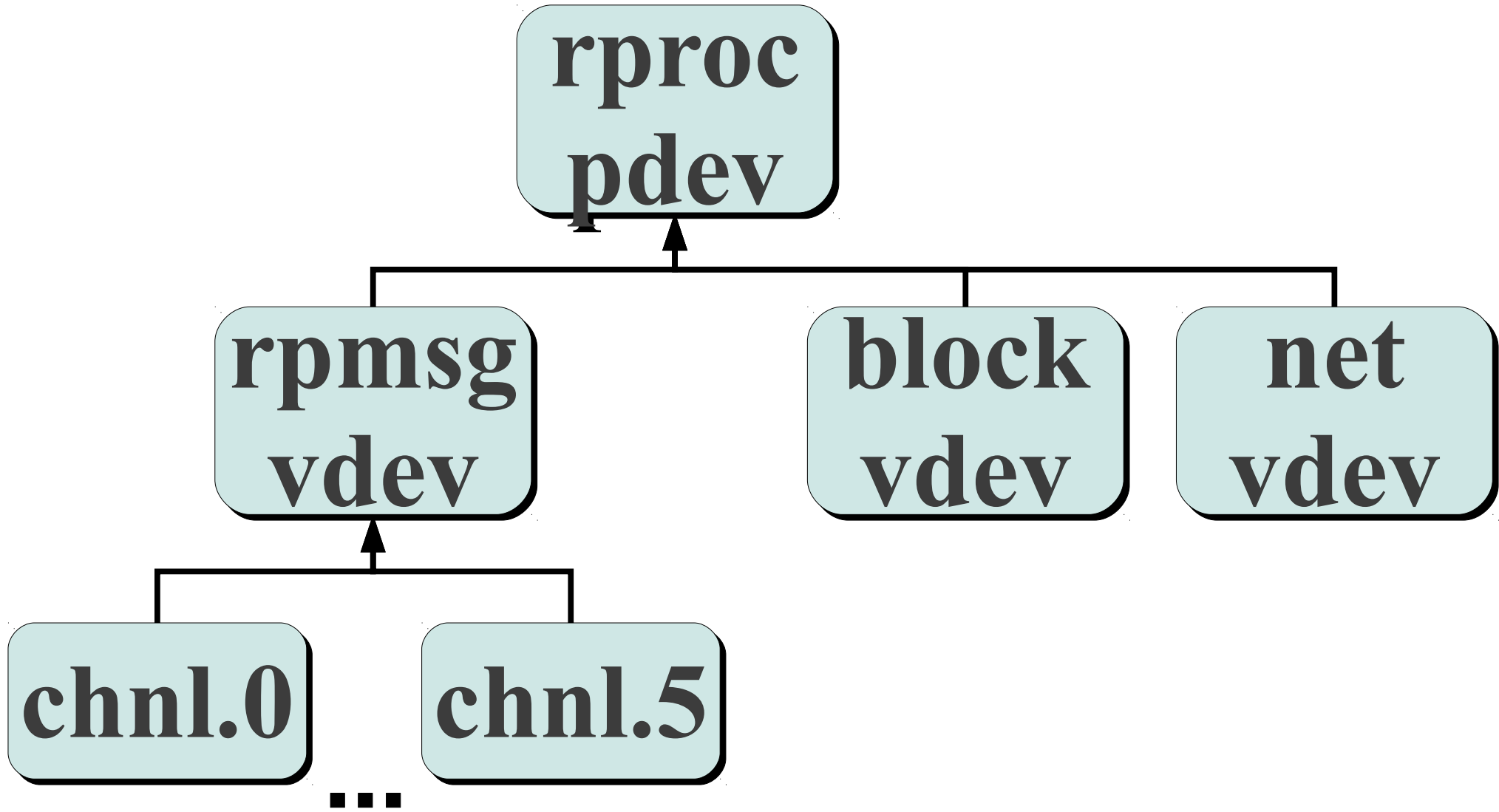
```
};
```



# rpmsg and virtio

- One vring per core
- Bus-owned buffers
- Messages begin with header
- `#define VIRTIO_ID_RPMSG 7`
- Dynamically create vdevs !  
(not implemented yet)

# Device topology



# Other goodies

device model

- **Name service for free !**
- **Crash ? Just remove the parent**
- **Runtime PM API**
- **DMA, CMA and IOMMU API**

OK, seems nice.

How do I add  
support for my  
xyz platform?

# xyz AMP support

```
static struct rproc_ops xyz_rproc_ops = {
    .start      = xyz_rproc_start,
    .stop       = xyz_rproc_stop,
    .kick       = xyz_rproc_kick,
};
```

```
static int __devinit xyz_rproc_probe(struct platform_device *pdev)
{
    struct xyz_rproc_pdata *pdata = pdev->dev.platform_data;
    struct xyz_rproc *my_proc;
    struct rproc *rproc;
    int ret;

    rproc = rproc_alloc(&pdev->dev, pdata->name, &xyz_rproc_ops,
                       pdata->firmware, sizeof(*my_proc));

    ... xyz-specific initialization ...

    ret = rproc_register(rproc);
```

# **xyz AMP support**

```
static int xyz_rproc_start(struct rproc *rproc)
{
    ... power on the core using xyz-specific stuff ...
}

static int xyz_rproc_stop(struct rproc *rproc)
{
    ... power off the core using xyz-specific stuff ...
}

static void xyz_rproc_kick(struct rproc *rproc, int vqid)
{
    ... tell core vqid has a msg ... xyz-specific stuff ...
}
```

# Er, firmware

- **Linux, RTOS or whatnot**
- **Put in /lib/firmware**
- **Only ELF (at this point)**
- **Special section: “.resource\_table”**
- **Specifies resource requirements**
- **Gets handled before boot**
- **Can also specify features**
- **E.g. virtio header resources**



# status

- **Platforms:**
  - **OMAP4**
  - **DaVinci real soonish (hopefully)**
  - **Other platforms in the works**
- **In the pipeline:**
  - **More VirtIO !**
  - **Socket interface**
  - **OMX offloading driver**
  - **Resource manager**
  - **...**



ohad@muesli: ~

```
ohad@muesli:~$ demo
```



Credits

**Brian Swetland**

**Suman Anna**  
**Fernando Lugo Guzman**  
**Iliyan Malchev**

**Mark Groesen**  
**G Anthony**

**Grant Likely**  
**Arnd Bergmann**  
**Rusty Russell**

# Thanks!

**Patches:**

**<http://git.kernel.org/.../ohad/remoteproc.git>**

**<http://git.kernel.org/.../ohad/rpmsg.git>**

**Slides:**

**[http://wizery.com/ohad/AMP\\_ELC2012.pdf](http://wizery.com/ohad/AMP_ELC2012.pdf)**