
Szedata v. 2

Jiri Slaby

slaby@liberouter.org

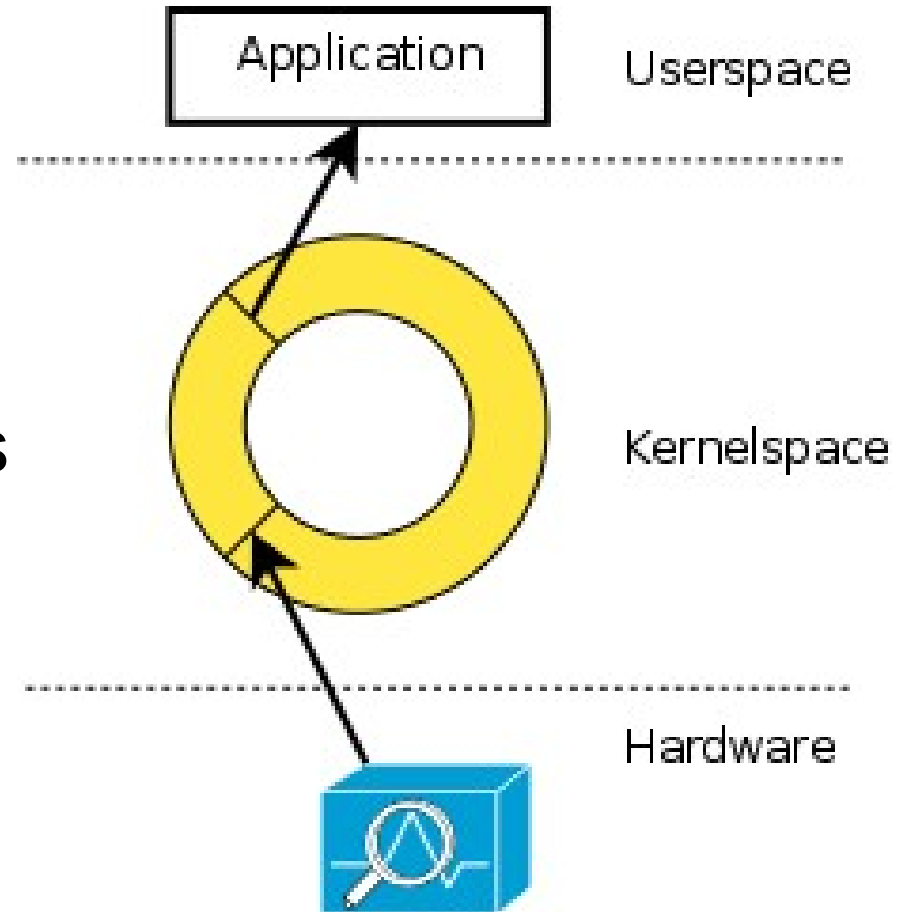


Introduction

- Fast transfers without copying data
- Szedata is that, so why szedata2?
 - New platform (NetCOPE)
 - \implies new concept, new design
 - Incompatible
 - no PPC
 - different data format
 - Similar approach in concept
 - mmap, subscribe, lock, unlock

Design

- RX, TX spaces
- N areas (interfaces)
- 2*N ring buffers
 - Wraps, tail=head
 - App sees only 2 spaces
- Two approaches
 - All data to all apps (different tasks)
 - Data divided among more apps (load balance)

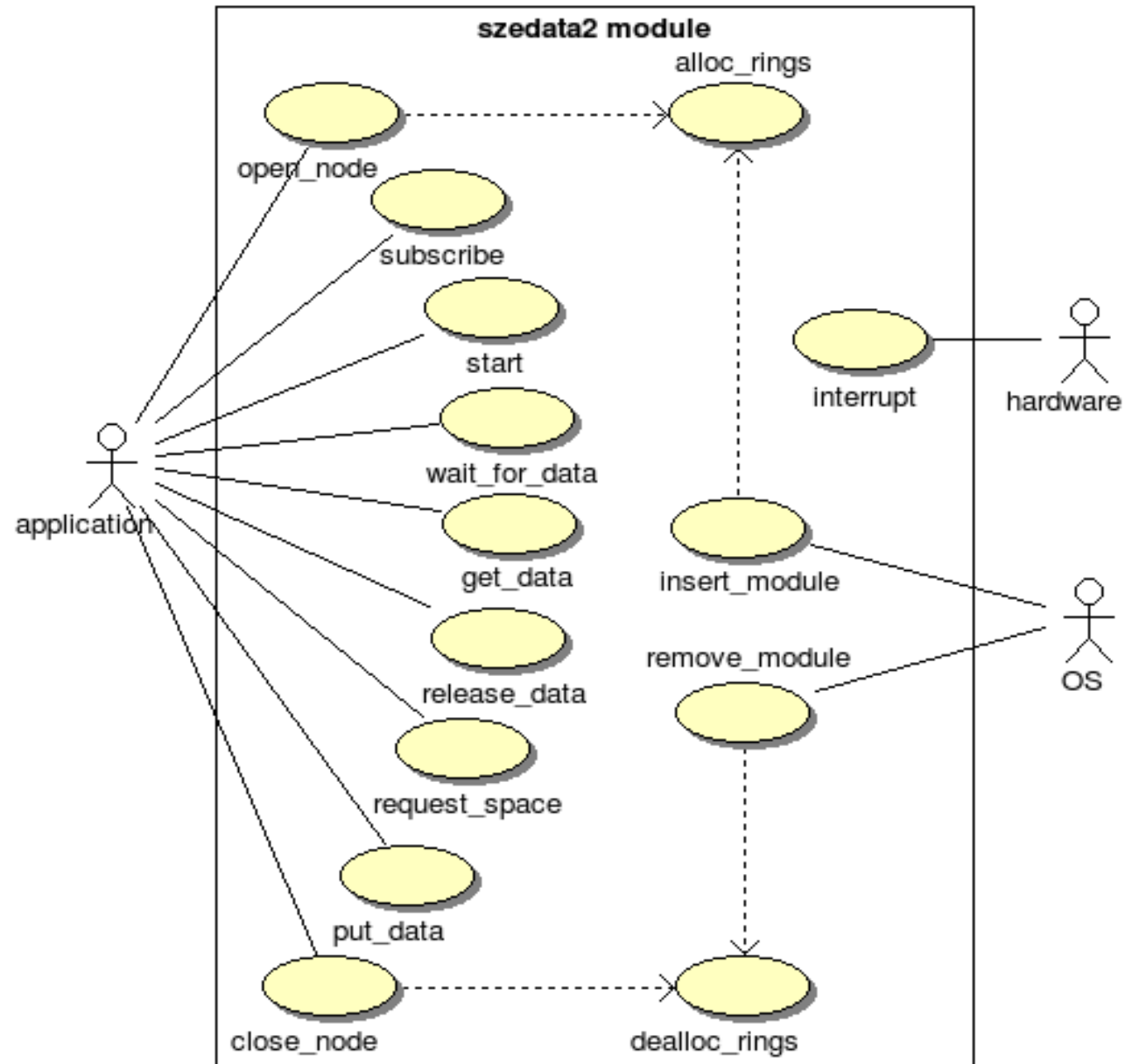


Design cont.

- **Each application wants all data**
- We need keep these pointers (per ring buffer):
 - Tail and head of HW
 - App RX: tail, head
 - App TX: only locked size
 - Only one writer at a time
- The slowest application might block others

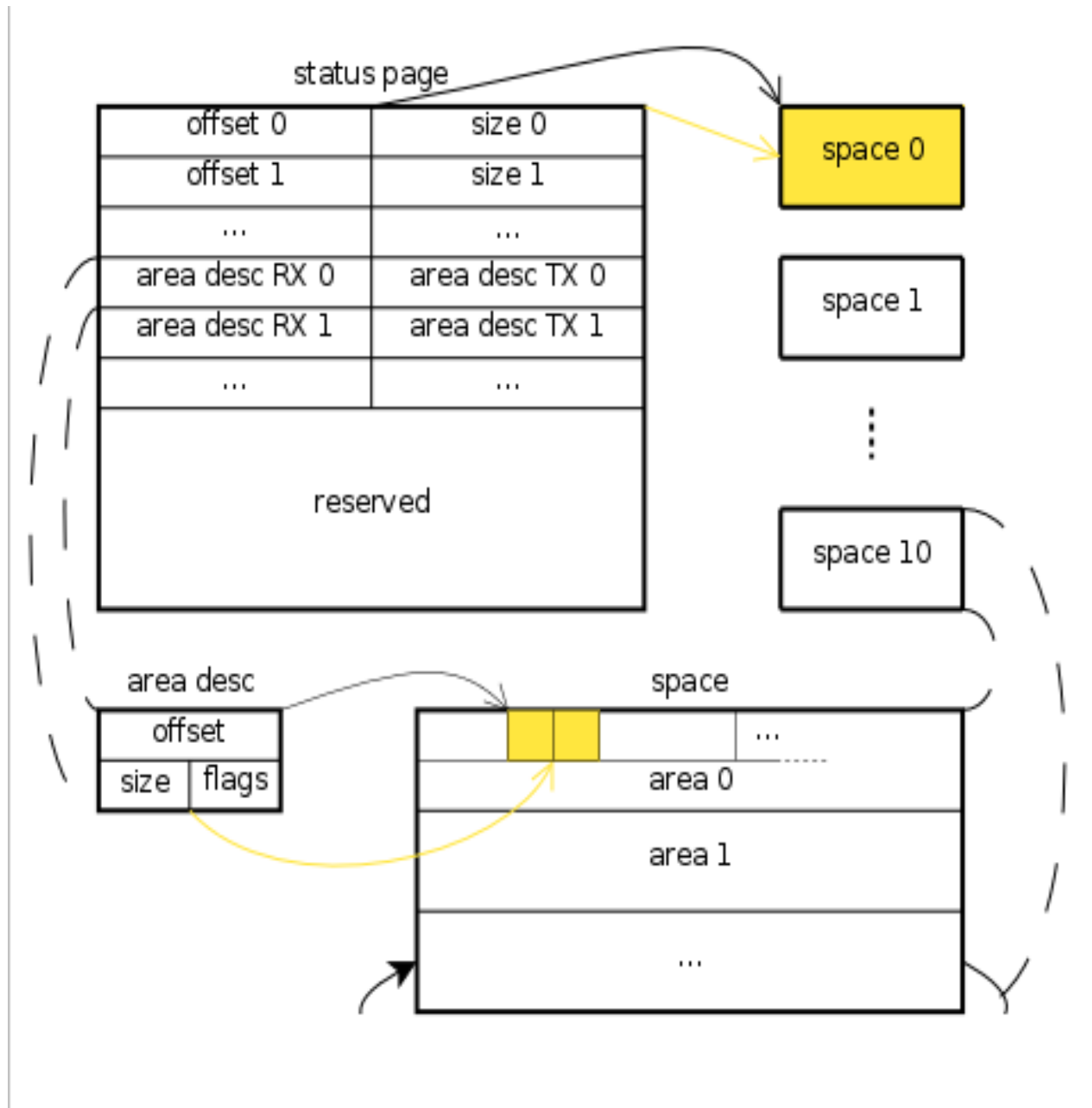
Interface

- Open
 - status_page
 - spaces
- Mmap
- Subscribe
- Start
- Repeat { Wait
- Lock
- Unlock }
- Close



Interface cont.

- Up to 11 spaces
 - RX, TX (areas)
 - Stats...
- Everything in status_page (depicted)
 - PAGE_SIZE anyway



Status

- Working non-HW dependant layer
 - Tested only on simulator – a module which **tries** to behave as the card, no known bugs :)
 - Libsze2 (see later)
 - Szetest2 – a testing application
- Design driver in few weeks
 - Fill in ring buffer descriptors
 - Setup DMA transfers
 - Start/stop HW
 - The rest is on the layer (counting pointers...)

Libsze2

- Unlike szedata, szedata2 provides its library
- Encapsulate init/deinit and processing
 - No more mmmaps, ioctls – abstraction
 - Easy to upgrade/change – dynamic library
- Code (SVN): `software/trunk/libs/libsze2`
- Docco:
http://andre.liberouter.org/~slaby/libsze2_api/

Libsze2 cont.

- Functions

```
struct szedata2 *szedata2_open(const char *node);
```

```
int szedata2_subscribe(struct szedata2 *sze2, __u32 *rx, __u32 *tx, __u32  
    rx_poll, __u32 tx_poll);
```

```
int szedata2_start(struct szedata2 *sze2);
```

```
int szedata2_poll(struct szedata2 *sze2, short *events, int timeout);
```

```
struct szedata2_lock *rx_lock_data(struct szedata2 *sze2, __u32 areas);
```

```
int rx_unlock_data(struct szedata2 *sze2, const struct szedata2_lock *lock);
```

```
struct szedata2_lock *tx_lock_data(struct szedata2 *sze2, __u32 size, __u8  
    area);
```

```
int tx_unlock_data(struct szedata2 *sze2, const struct szedata2_lock *lock,  
    __u32 size);
```

```
void szedata2_close(struct szedata2 *sze2);
```

End



Thanks for your attention.

Questions?