Dynamic mbuf

THOMAS MONJALON – MELLANOX
OLIVIER MATZ – 6WIND

Features require metadata

- offloads in NIC
  - load balancing (flow steering)
  - segmentation (LRO, TSO)
  - checksums
  - classification
  - tunneling, inline protocol processing (IPsec, NVMe)
- lookaside or inline processing
  - crypto symmetric/asymmetric
  - lossless compression/decompression (stateless or stateful)
  - pattern matching

- Note: software emulation can fill some gaps

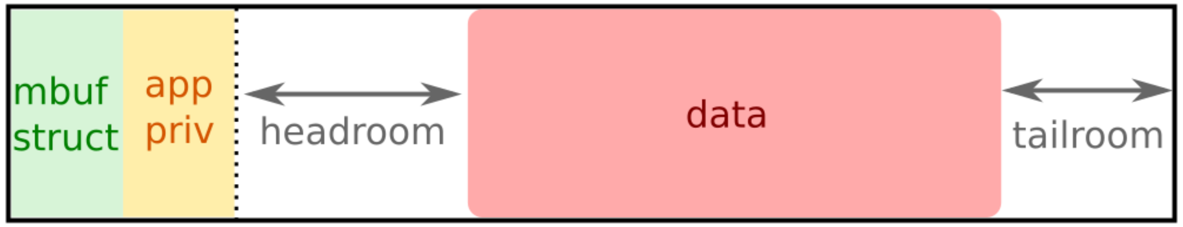
struct rte_mbuf

- Metadata for a network packet segment

- **Data** size, pointer (virtual and IOVA), private data size, external buffer metadata pointer
- **Segment** size, total count and pointer to next
- **Protocol** data (packet type, layer sizes, tunnels, checksums, VLAN, LRO, TSO, IPsec...)
- Flow **classification** (port id, queue id, hash, traffic class...)
- Timestamp, PTP
- User metadata
- **Offload** flags

Private Data for Applications

- Space can be reserved on mempool allocation



- Application configures mempool
- Transparent for DPDK

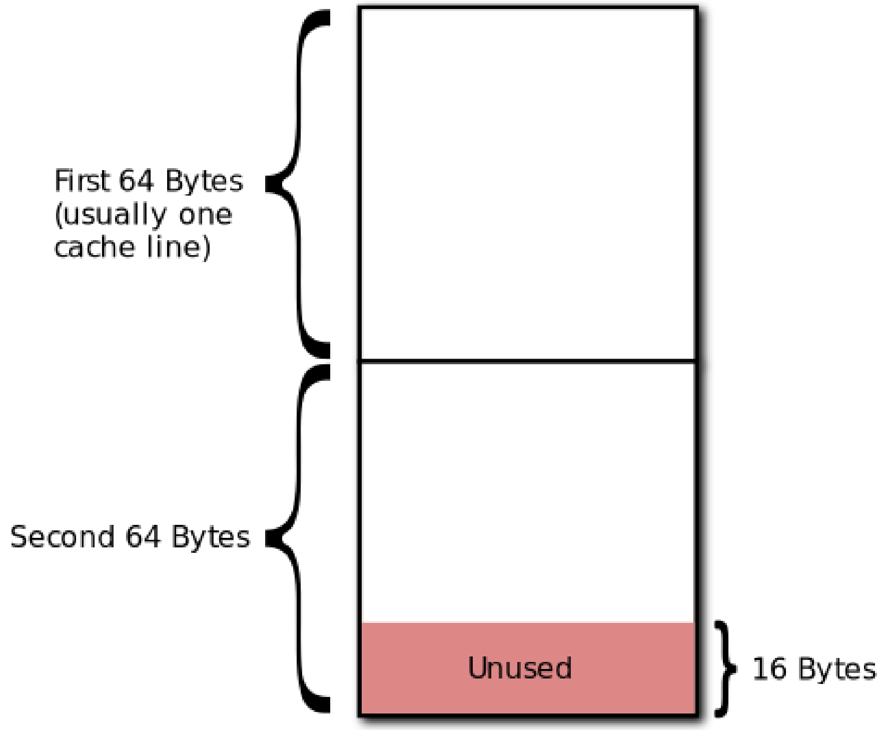
Current mbuf Limitations

Limited Space

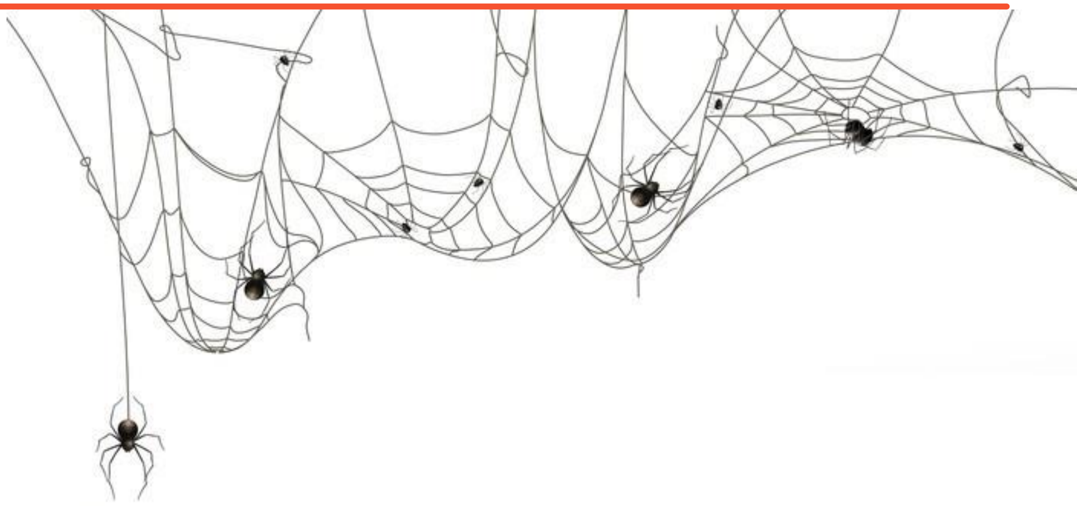
- Small mbuf == Less cache misses

- Only 2 cache lines
  - 2 x 64 = **128** Bytes
  - depends on architecture

- Last free space
  - pahole finds **16** Bytes at the end



Wasted Space



For one application,

For one use case,

Some mbuf fields are not used.

Some features are **rarely** used.

Mutually Exclusive Features

- Long term, features using the same bytes will clash
  - **Placeholders** with vague description are **bad**

    `seqn, tx_metadata, userdata, usr`

  - **Unions** of separate features are **bad**

32-bit	RSS	FDIR low	sched queue		user tag (distributor)	Tx metadata
32-bit		FDIR high	sched class + color	eventdev Tx queue		

Stability

- Removal or move in mbuf is a strong **ABI** break

- **Vector** implementations are tied to mbuf layout

- Slow evolution
- Target: no layout change at all in future

Elsewhere

- Same issue in Linux XDP
  - http://vger.kernel.org/netconf2019_files/xdp-metadata-discussion.pdf
- FreeBSD m_tag
  - https://www.freebsd.org/cgi/man.cgi?query=mbuf_tags

DPDK

DATA PLANE DEVELOPMENT KIT

Extend with Dynamic Fields

Why not Allocating External Structure?

- Flexible
  - any length
  - chained
- Performance impact
  - allocate / free
  - cache miss
- Needs specific pools

Why not Increasing Size?

- Simple
- Performance impact
- Does not avoid ABI breakage each time layout is changed
- Space is still wasted (many unused fields)

Why not Selective Layout?

Application would choose between different mbuf layouts depending on its needs

- Requires as many structures as use cases
- Difficult to adapt and optimize drivers for all possible layouts

Design of Dynamic Fields

- **Register**
  - on demand, depending on use case
  - unused fields don't use space in mbuf

name
size
alignment
flags

$\Rightarrow$ offset

- Drivers and applications access to a **dynamic offset** in the mbuf
  - small performance impact
- System-wide
  - impacts all mbufs in all pools
- Same logic for **dynamic bits** in offload flags

name
count

$\Rightarrow$ bit number

API

```
const struct rte_mbuf_dynfield rte_mbuf_dynfield_my_feature = {
        .name = "rte_mbuf_dynfield_my_feature",
        .size = sizeof(uint64_t),
        .align = __alignof__(uint64_t),
        .flags = 0,
};
```

Register the field

```
offset = rte_mbuf_dynfield_register(&dynfield);
if (offset < 0)
     /* error */
```

Read/Write the field

```
*RTE_MBUF_DYNFIELD(mbuf, offset, uint64_t *) = 0x1337beef;
```

Example of Field

- Helper to register flag and field together
  - `rte_mbuf_dyn_timestamp_register()`

- Feature-specific accessors
  - `rte_mbuf_dyn_timestamp_get(mbuf)`
  - `rte_mbuf_dyn_timestamp_set(mbuf, value)`
  - `rte_mbuf_dyn_timestamp_avail(mbuf)`

Drawbacks / Limitations

- Lower **performance** than accessing a static field
  - Early benchmark:

    +2 cycles for write access
    +3 cycles for read access

- Cannot unregister dynamic fields

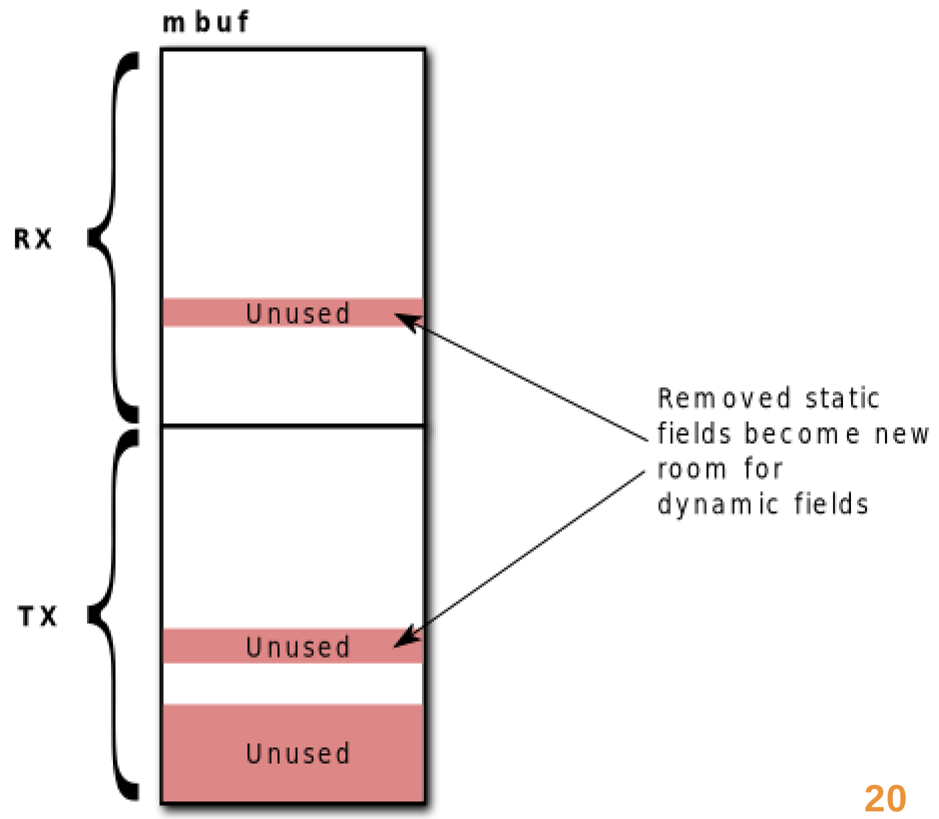
- No magic: space is still limited (but more flexible)

DPDK
DATA PLANE DEVELOPMENT KIT

Get even More Space

Plan for Future

- **Sustainable** if enough space to combine a lot of features
- **Convert** some fields from static to dynamic
- Would add room in Rx (first) cache line
    - performance gain for fields moved in Rx part
    - registration flags to choose the cache line



Criteria for Dynamic Field

- Uncommon use
- Vendor-specific
- Performance degradation by a couple of cycles not critical
- Union'ed (exclusive) feature

Remove User Data

- mbuf field (in second half)
  - `void *userdata`
  - `uint64_t udata64`
- Application can register its own well-defined field

Remove User Tag

- mbuf field (union'ed in first half)
  - `uint32_t usr`

- Cannot be used together with RSS hash

- Used only by distributor library
  - could use a well-defined dynamic field

Convert External Buffer Data Pointer

- mbuf field (in second half)
  - `struct rte_mbuf_ext_shared_info *shinfo`

- Accessed only on external buffer attach

- Part of mbuf API
  - Difficult to convert

Convert PTP

- Offload flags
  - `PKT_RX_IEEE1588_PTP`
  - `PKT_RX_IEEE1588_TMST`
  - `PKT_TX_IEEE1588_TMST`
- mbuf field (in second half)
  - `uint16_t timesync`

- IEEE1588 PTP is a payload on top of UDP
- Why is it part of mbuf API?

Convert Timestamp

- mbuf field (in first half)
  - `uint64_t timestamp`

- Not performance critical?
- Not widely used
- In first half (Rx part)

Convert Sequence Number

- mbuf field (in second half)
  - `uint32_t seqn`

- Not enough defined
- Not widely used

Convert Hierarchical Scheduler

- mbuf sub-struct (union'ed in first half)
  - `uint32_t queue_id`
  - `uint8_t traffic_class`
  - `uint8_t color`

- Feature union'ed with RSS
- QoS not always done

Convert eventdev Tx Adapter

- mbuf field (union'ed in first half)
  - `uint16_t txq`

- Feature union'ed with RSS
- eventdev not always in use

Convert Tx Metadata

- mbuf field (union'ed in first half)
  - `uint32_t tx_metadata`

- Feature union'ed with RSS
- Application-specific usage

More?

Other fields could be discussed.

The conversion may be a long way
happening as jumps when ABI breakage window is open.

Conclusion

TO REVIEW (for 19.11)

Add dynamic mbuf API.

TODO (for 20.11)

Migrate some static fields to dynamic.

DPDK
DATA PLANE DEVELOPMENT KIT

Questions?