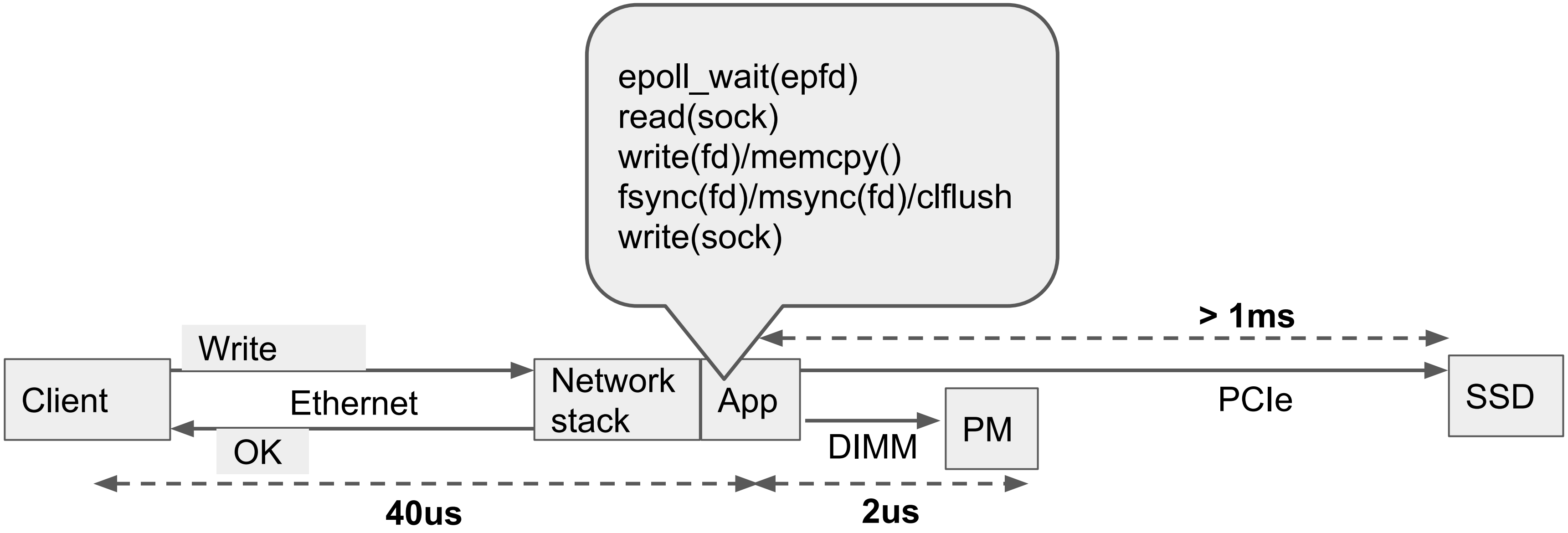
Extending Linux Network Stack for Persistent Memory

Michio Honda (NEC Laboratories Europe)
Giuseppe Lettieri (Universita di Pisa)
Lars Eggert (NetApp)
Douglas Santry (NetApp)

Problem

- Persistent memory (PM) is emerging
  - HPE NVDIMM, Intel 3D-Xpoint etc.
- **End-to-end latencies are now dominated by networking**

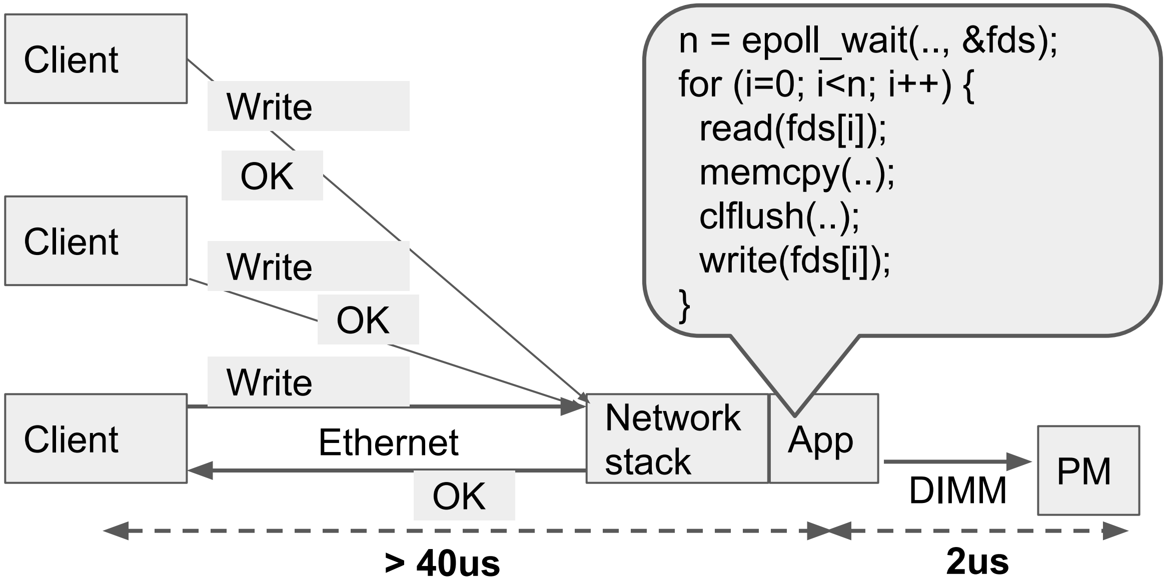


epoll_wait(epfd)
read(sock)
write(fd)/memcpy()
fsync(fd)/msync(fd)/clflush
write(sock)

Numbers and graphs are from *Honda et al., “PASTE: Network Stacks Must Integrate with NVMM Abstractions”, ACM HotNets 2016

Problem

- Persistent memory (PM) is emerging
  - HPE NVDIMM, Intel 3D-Xpoint etc.
- **End-to-end latencies are now dominated by networking**



```
n = epoll_wait(.., &fds);
for (i=0; i<n; i++) {
  read(fds[i]);
  memcpy(..);
  clflush(..);
  write(fds[i]);
}
```

Graph here

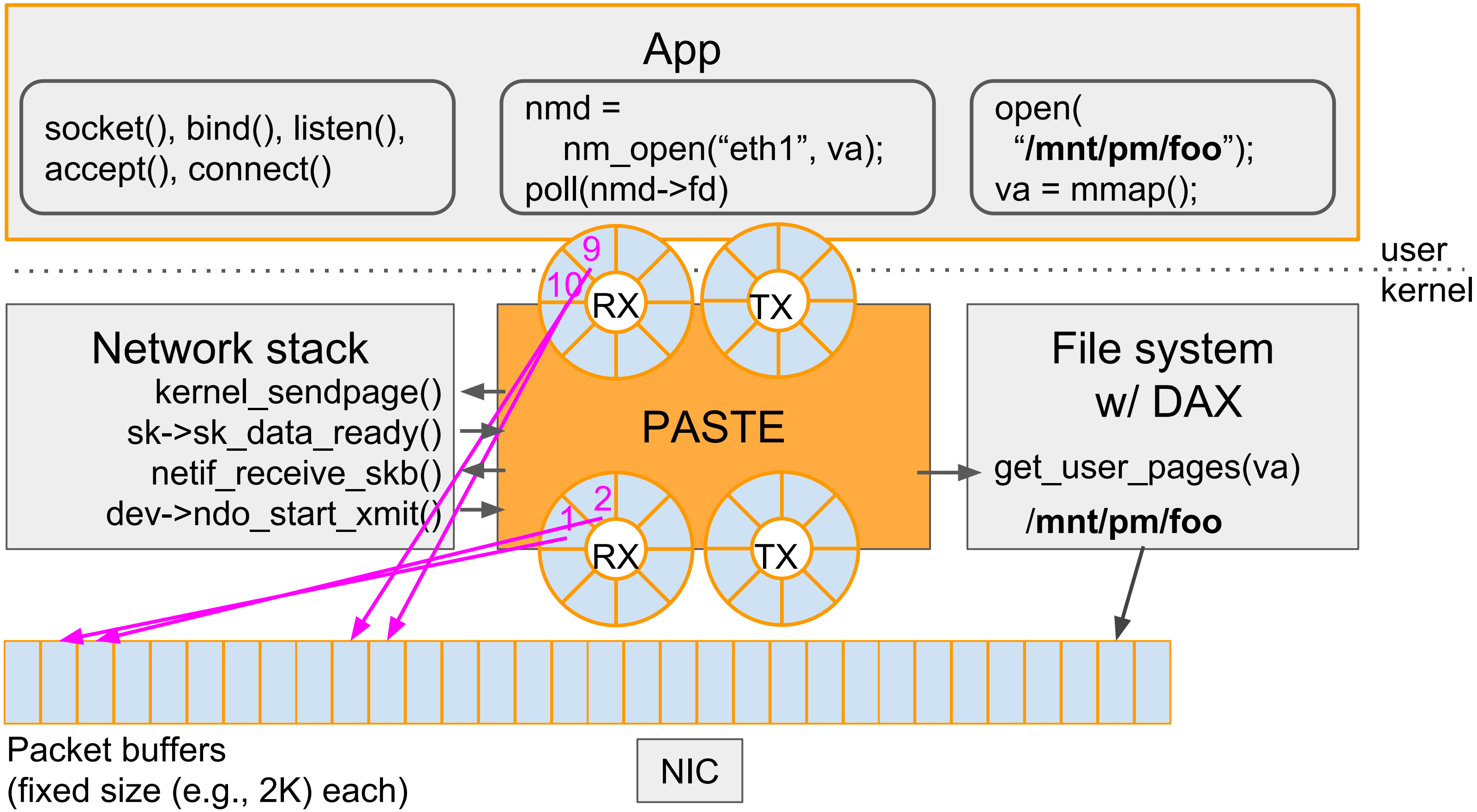
Numbers and graphs are from *Honda et al., “PASTE: Network Stacks Must Integrate with NVMM Abstractions”, ACM HotNets 2016

Status Quo

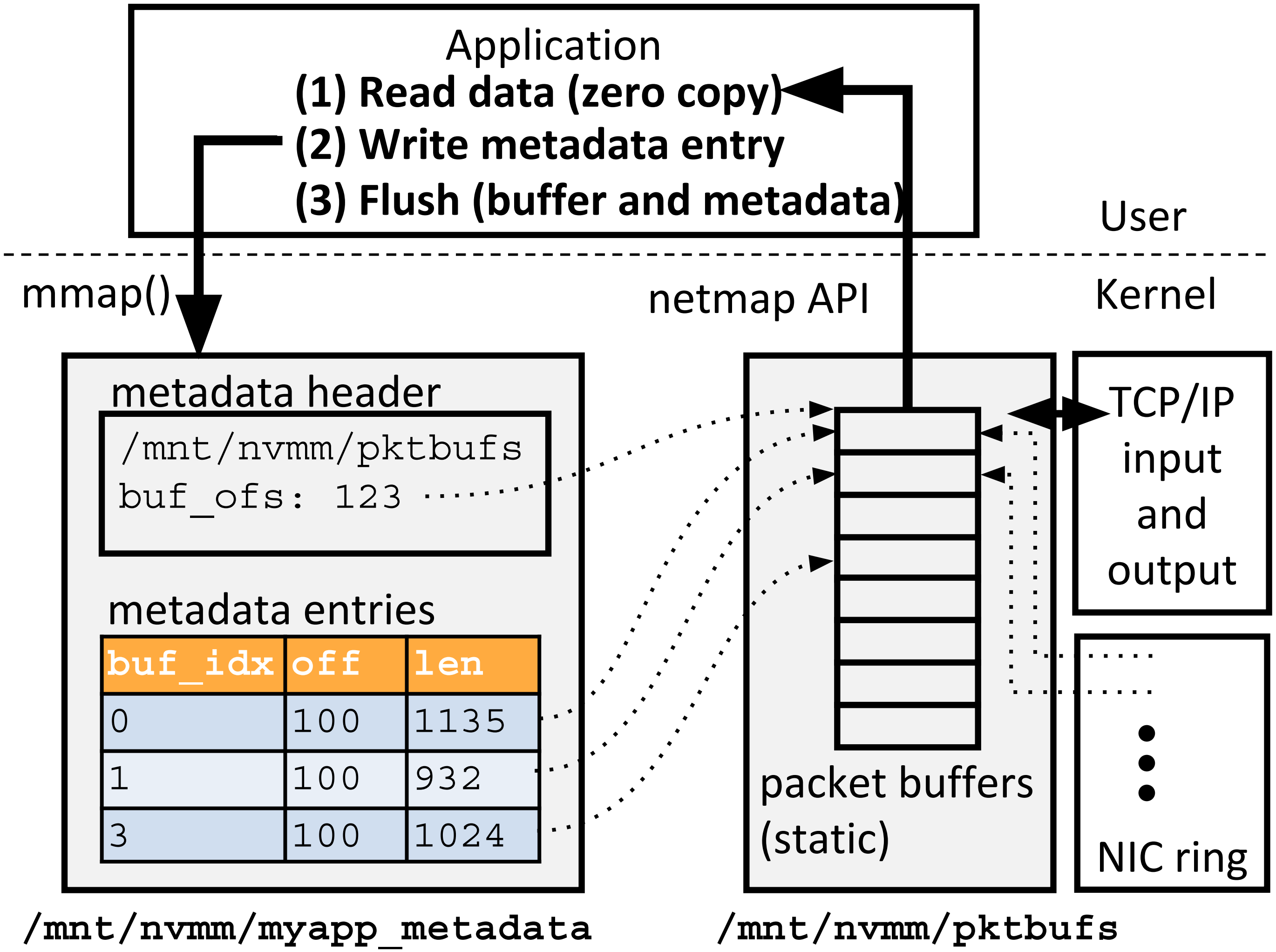
	Zero copy	Syscall batching	Synchronous I/O	Linux TCP/IP	Named packet buffers
MSG_ZEROCOPY	✔	x	x	✔	x
KCM	x	✔	x	✔	x
DPDK	✔	✔	✔	x	x
PASTE	✔	✔	✔	✔	✔

PASTE Design

- Zero-copy packet I/O to and from PM-backed file
- All the best practices for high-performance network stacks



In Action



Application

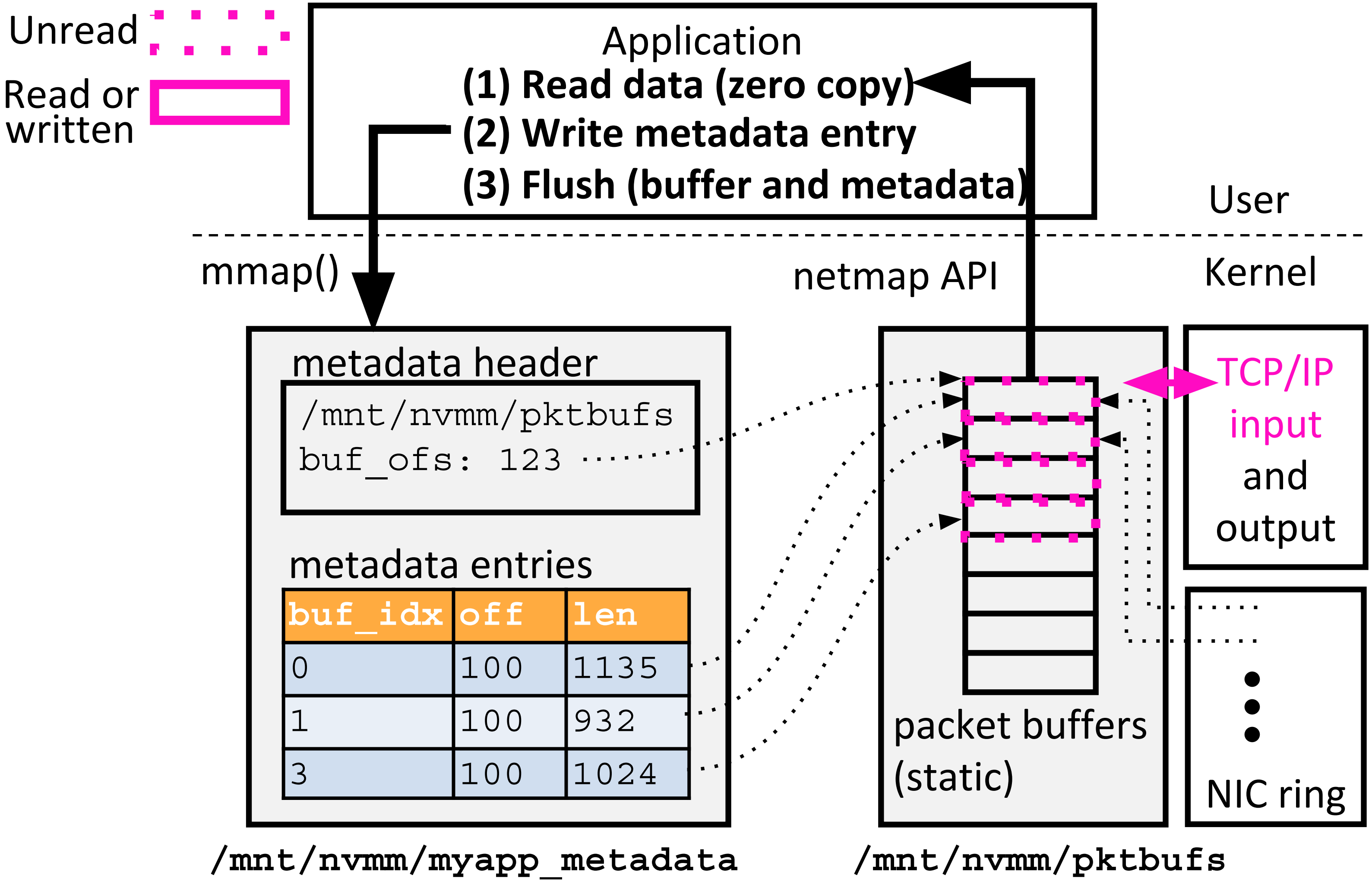
(1) Read data (zero copy)

(2) Write metadata entry

(3) Flush (buffer and metadata)

buf_idx	off	len
0	100	1135
1	100	932
3	100	1024

In Action



Unread

Read or written

Application

(1) Read data (zero copy)

(2) Write metadata entry

(3) Flush (buffer and metadata)

User

Kernel

mmap()

netmap API

metadata header

```
/mnt/nvmm/pktbufs
buf_ofs: 123
```

metadata entries

buf_idx	off	len
0	100	1135
1	100	932
3	100	1024

packet buffers (static)

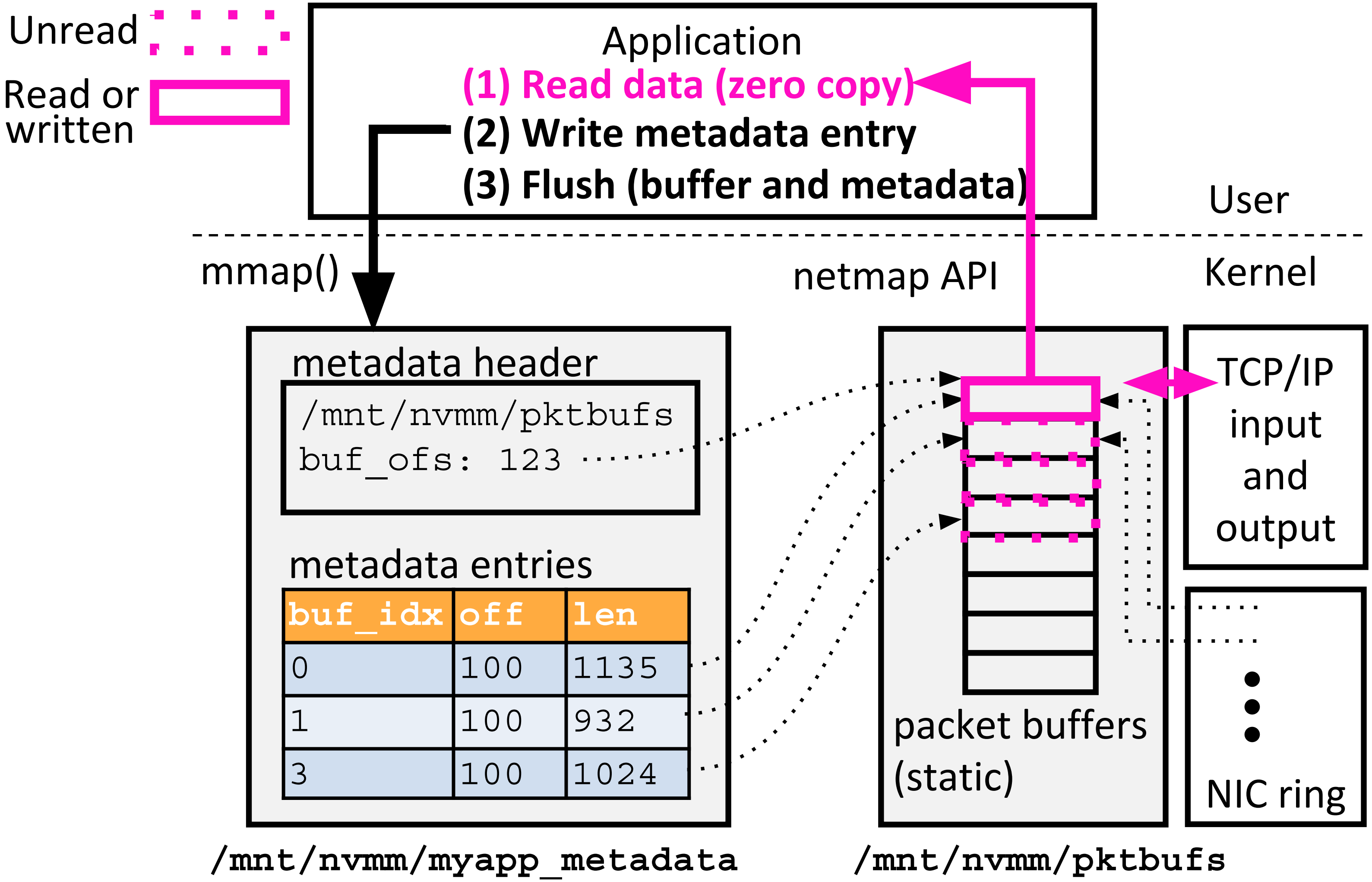
TCP/IP input and output

NIC ring

/mnt/nvmm/myapp_metadata

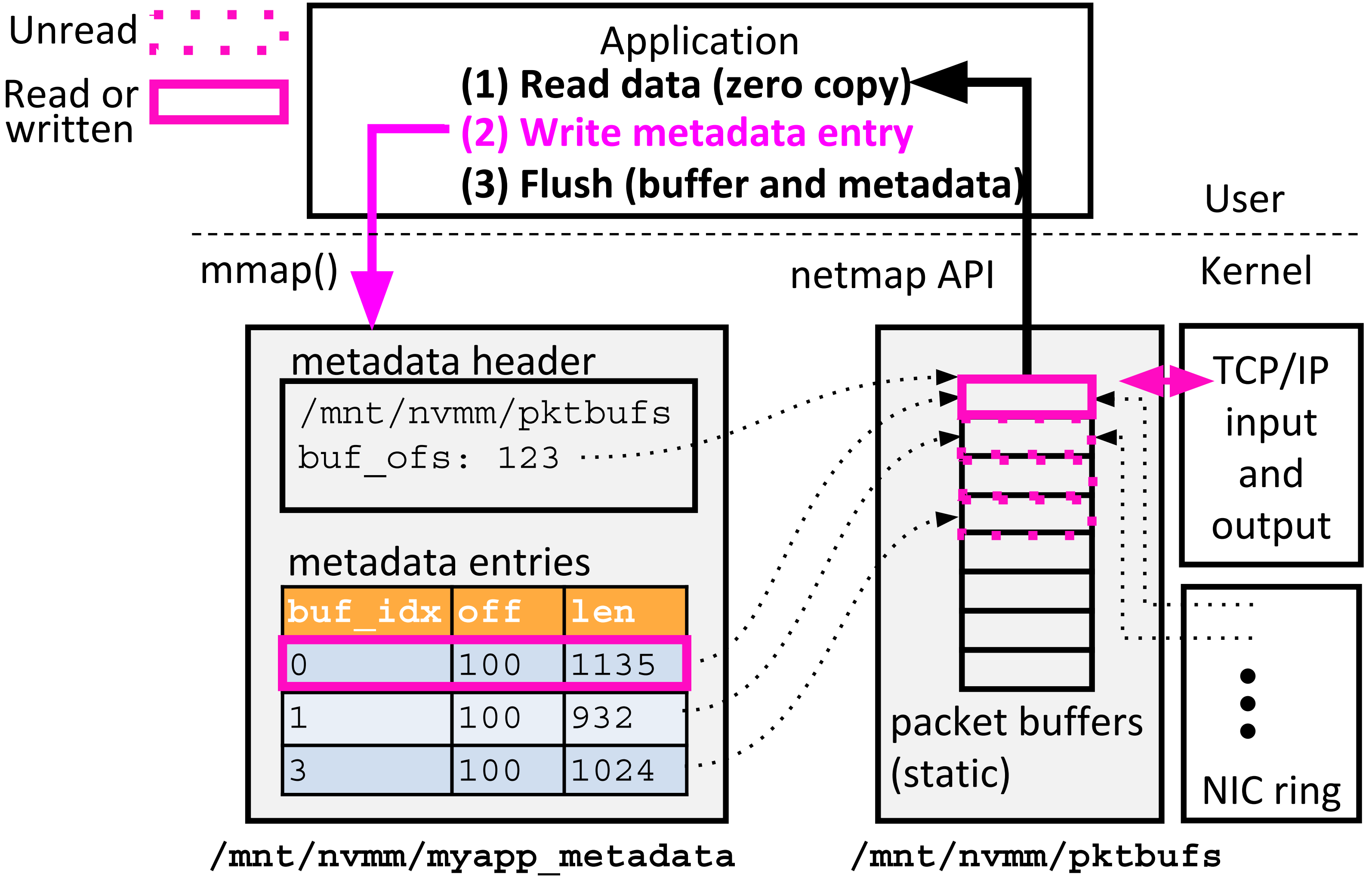
/mnt/nvmm/pktbufs

In Action



buf_idx	off	len
0	100	1135
1	100	932
3	100	1024

In Action



Unread

Read or written

Application

(1) Read data (zero copy)

(2) Write metadata entry

(3) Flush (buffer and metadata)

User

Kernel

mmap()

netmap API

metadata header

```
/mnt/nvmm/pktbufs
buf_ofs: 123
```

metadata entries

buf_idx	off	len
0	100	1135
1	100	932
3	100	1024

packet buffers (static)

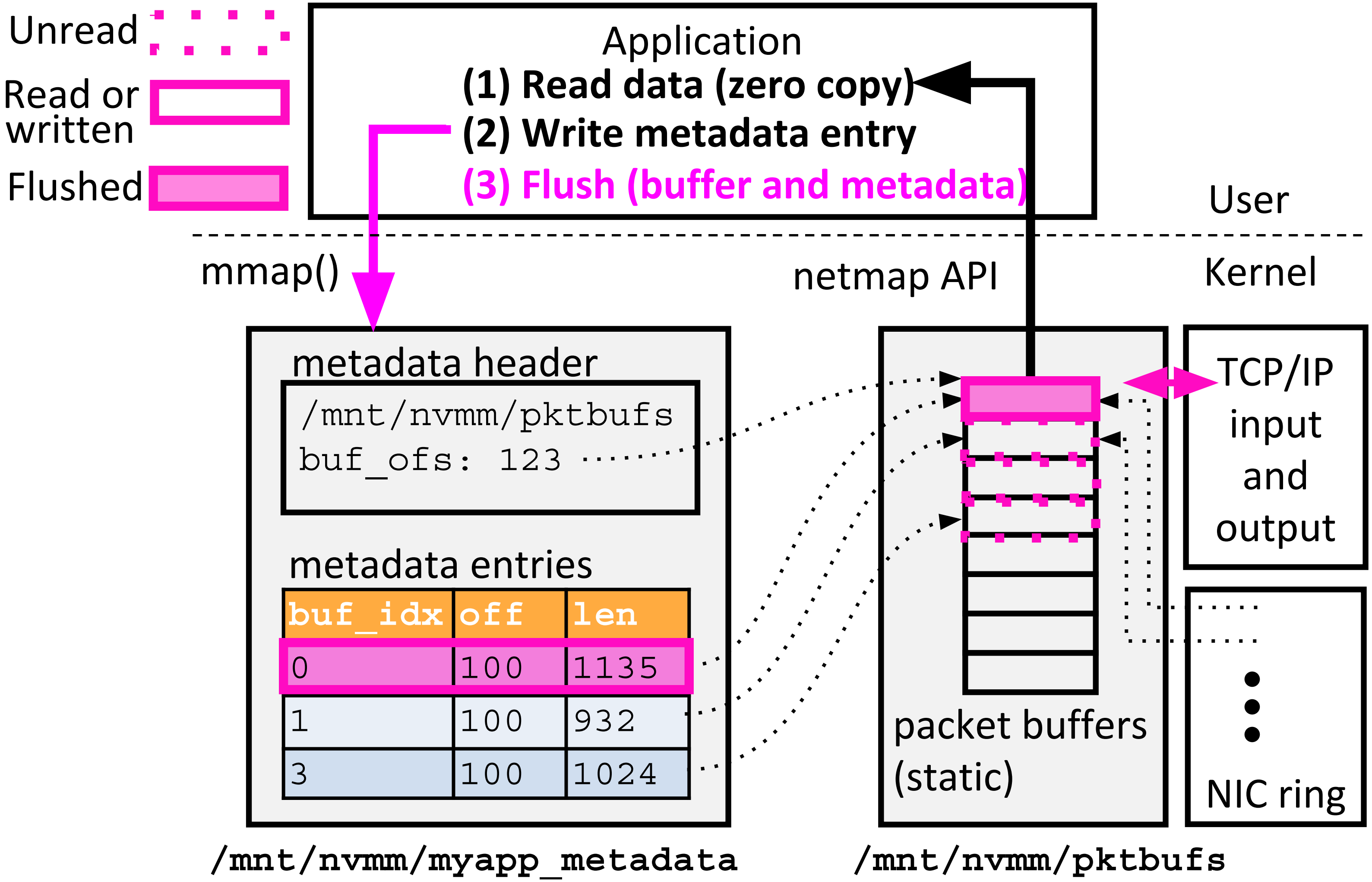
TCP/IP input and output

NIC ring

/mnt/nvmm/myapp_metadata

/mnt/nvmm/pktbufs

In Action



Unread

Read or written

Flushed

Application

(1) Read data (zero copy)

(2) Write metadata entry

(3) Flush (buffer and metadata)

User

Kernel

mmap()

netmap API

metadata header

```
/mnt/nvmm/pktbufs
buf_ofs: 123
```

metadata entries

buf_idx	off	len
0	100	1135
1	100	932
3	100	1024

packet buffers (static)

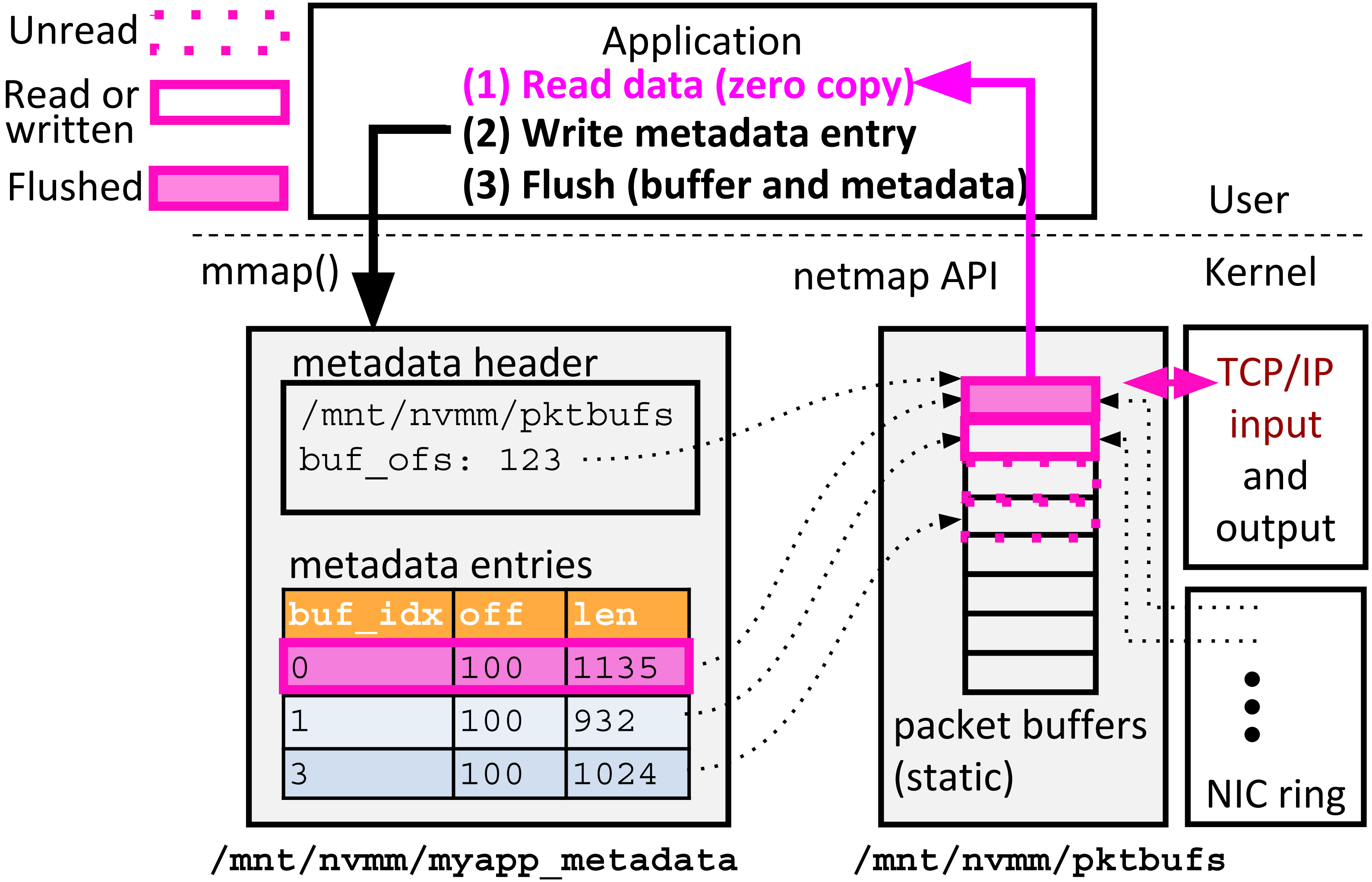
TCP/IP input and output

NIC ring

/mnt/nvmm/myapp_metadata

/mnt/nvmm/pktbufs

In Action



Unread

Read or written

Flushed

Application

(1) Read data (zero copy)

(2) Write metadata entry

(3) Flush (buffer and metadata)

User

Kernel

mmap()

netmap API

metadata header

/mnt/nvmm/pktbufs
buf_ofs: 123

metadata entries

buf_idx	off	len
0	100	1135
1	100	932
3	100	1024

packet buffers (static)

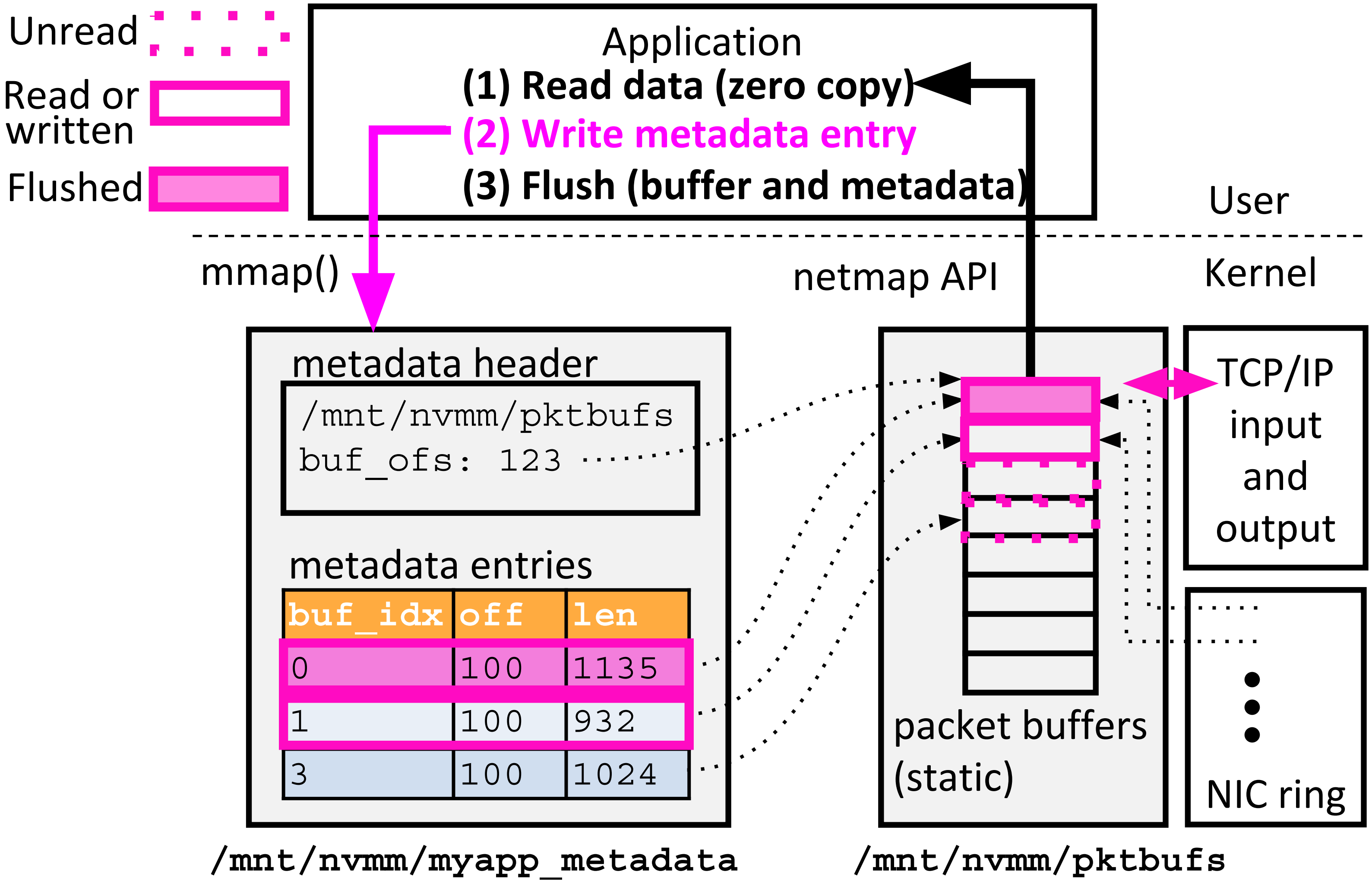
TCP/IP input and output

NIC ring

/mnt/nvmm/myapp_metadata

/mnt/nvmm/pktbufs

In Action



Unread

Read or written

Flushed

Application

(1) Read data (zero copy)

(2) Write metadata entry

(3) Flush (buffer and metadata)

User

Kernel

mmap()

netmap API

metadata header

```
/mnt/nvmm/pktbufs
buf_ofs: 123
```

metadata entries

buf_idx	off	len
0	100	1135
1	100	932
3	100	1024

packet buffers (static)

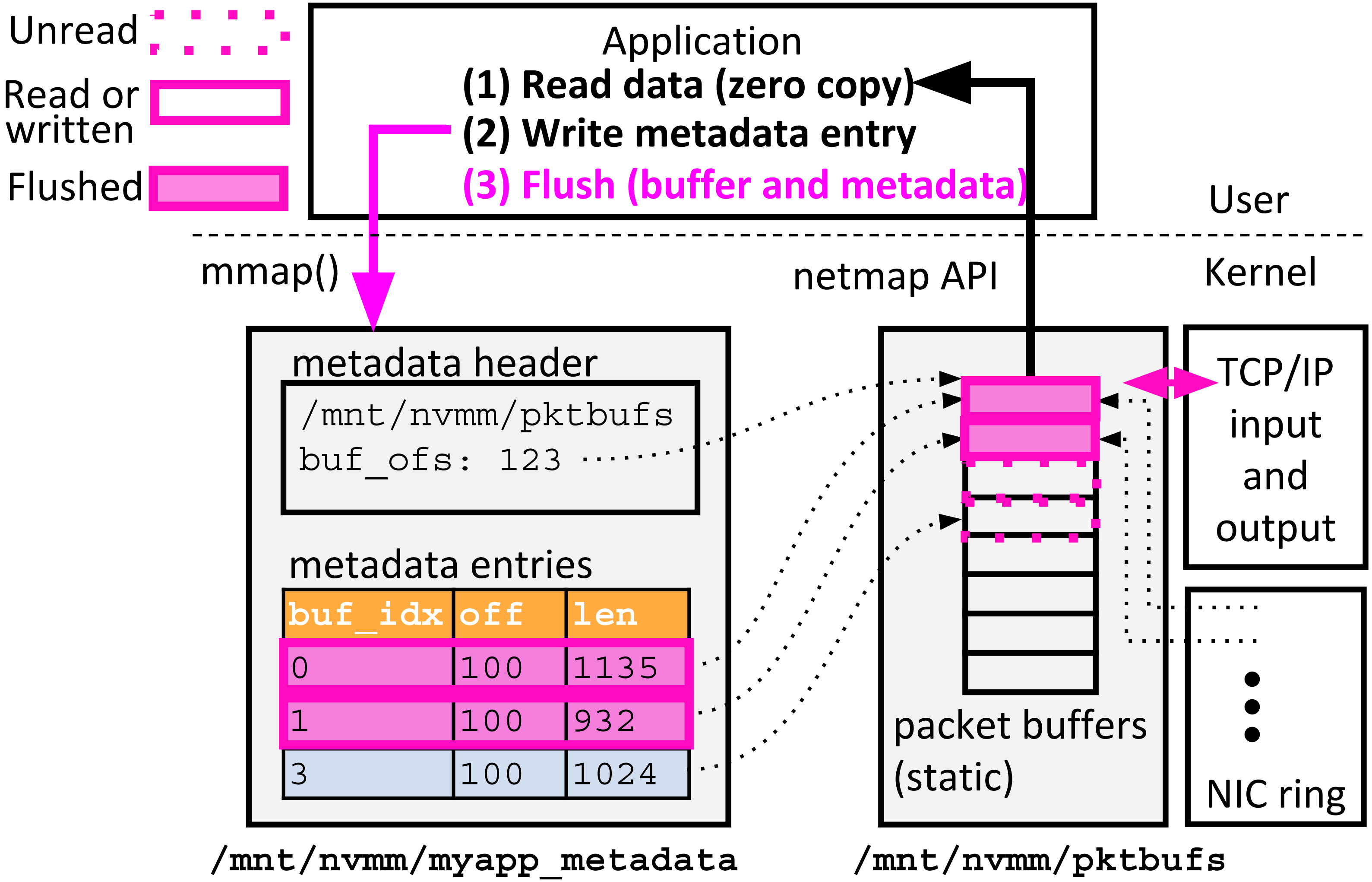
TCP/IP input and output

NIC ring

/mnt/nvmm/myapp_metadata

/mnt/nvmm/pktbufs

In Action



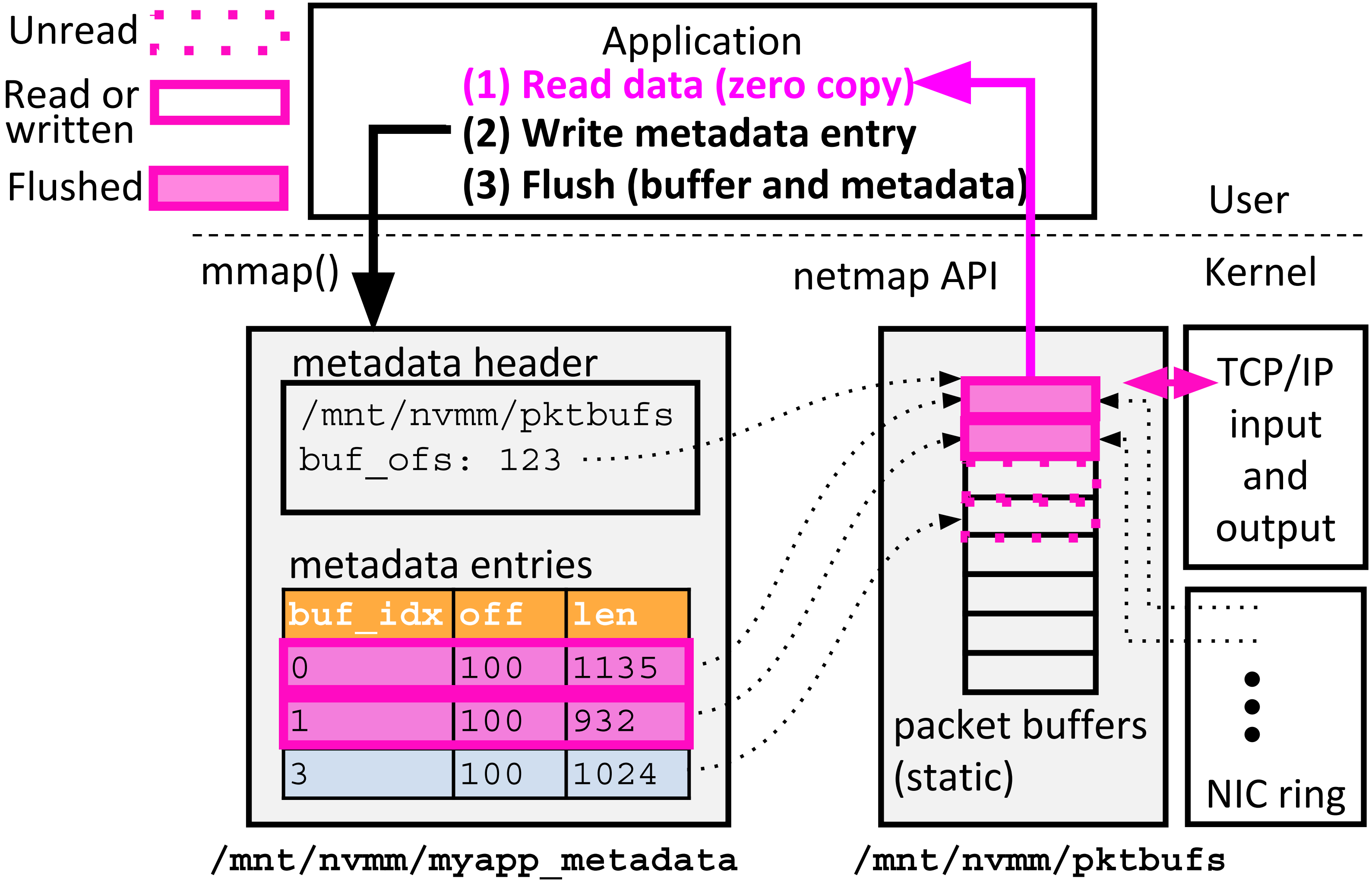
(1) Read data (zero copy)

(2) Write metadata entry

(3) Flush (buffer and metadata)

buf_idx	off	len
0	100	1135
1	100	932
3	100	1024

In Action



Unread

Read or written

Flushed

Application

(1) Read data (zero copy)

(2) Write metadata entry

(3) Flush (buffer and metadata)

User

Kernel

mmap()

netmap API

metadata header

```
/mnt/nvmm/pktbufs
buf_ofs: 123
```

metadata entries

buf_idx	off	len
0	100	1135
1	100	932
3	100	1024

packet buffers (static)

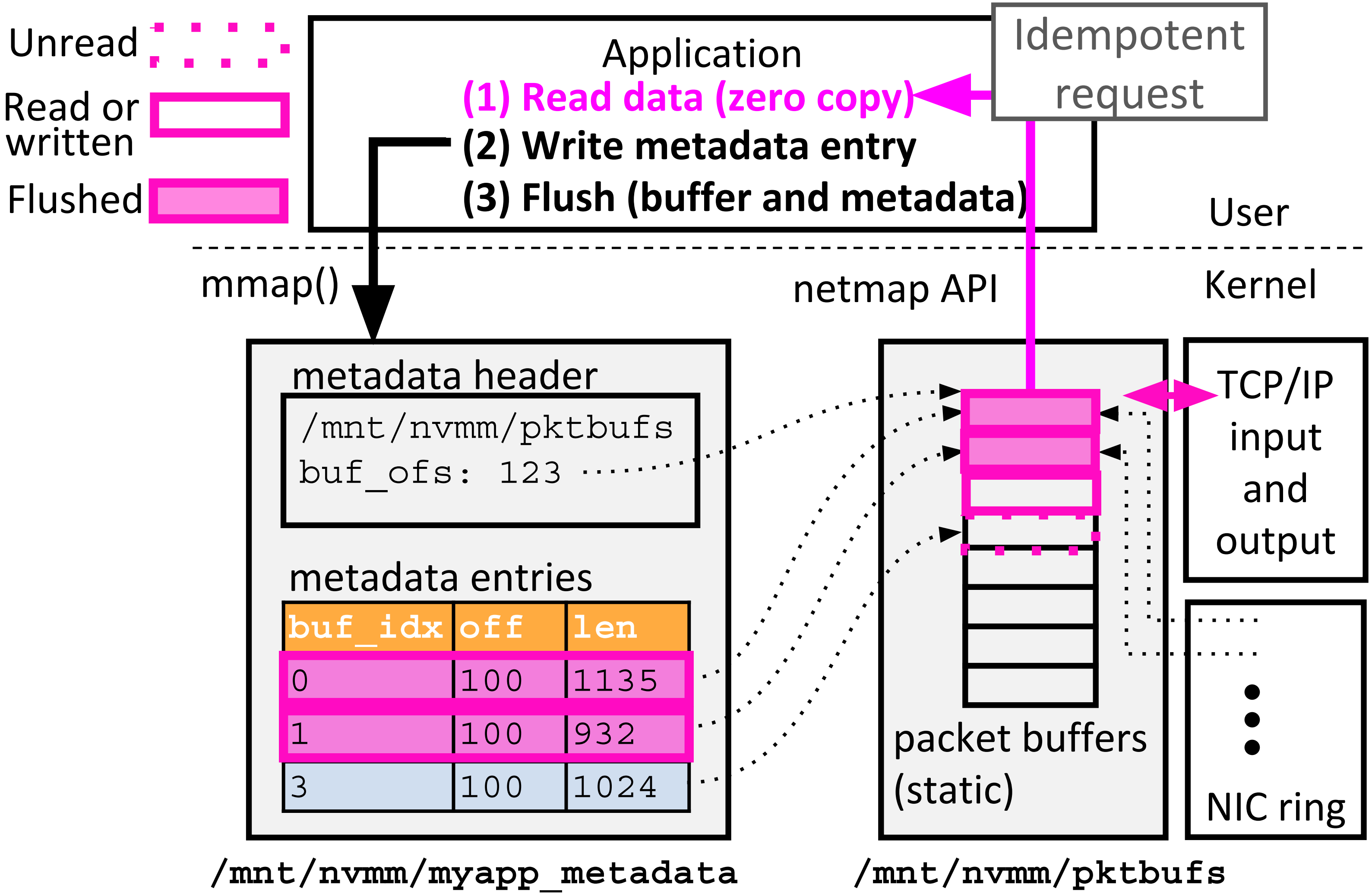
TCP/IP input and output

NIC ring

`/mnt/nvmm/myapp_metadata`

`/mnt/nvmm/pktbufs`

In Action



Unread

Read or written

Flushed

Application

(1) Read data (zero copy)

(2) Write metadata entry

(3) Flush (buffer and metadata)

Idempotent request

User

Kernel

mmap()

netmap API

metadata header

```
/mnt/nvmm/pktbufs
buf_ofs: 123
```

metadata entries

buf_idx	off	len
0	100	1135
1	100	932
3	100	1024

packet buffers (static)

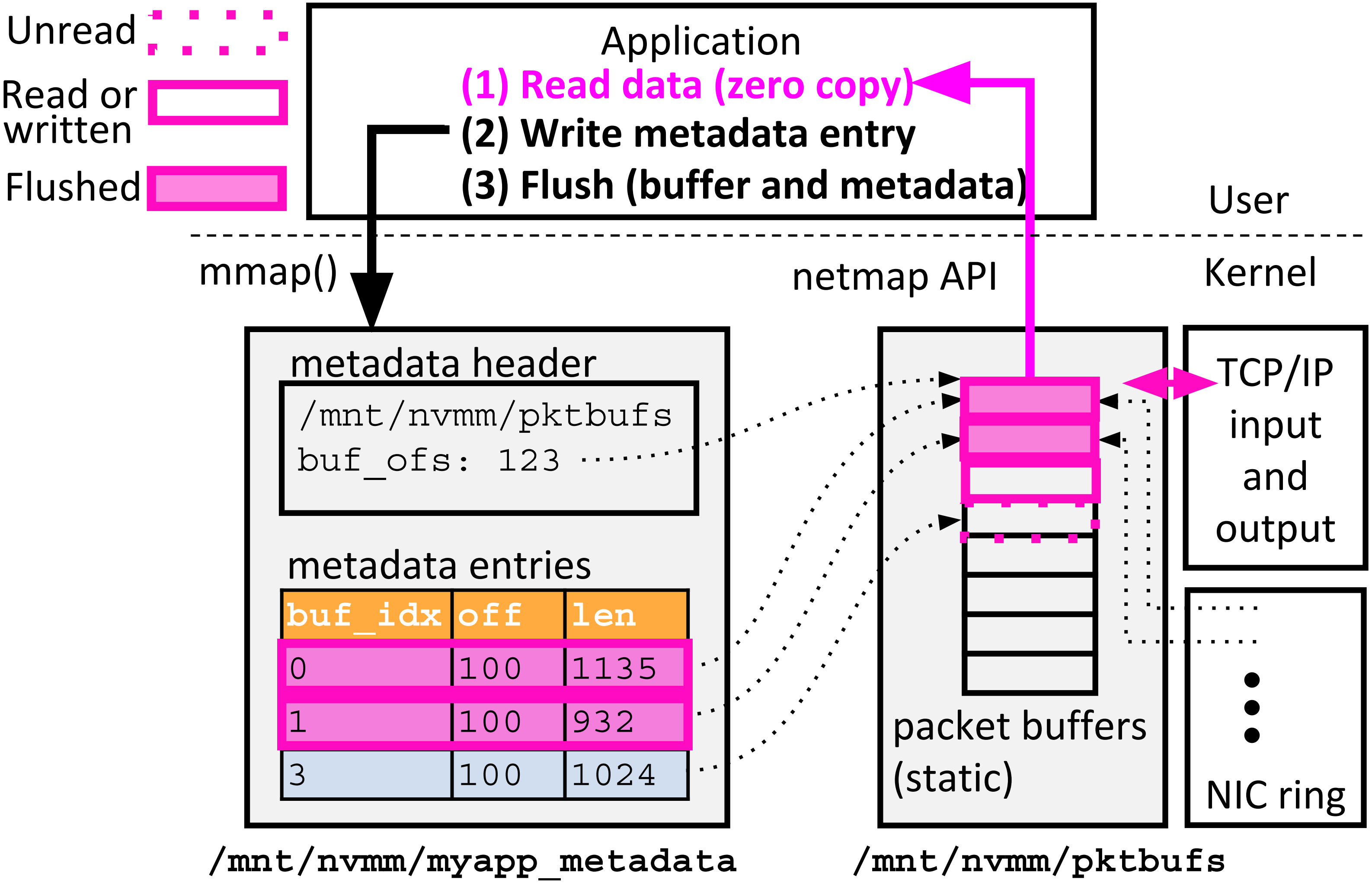
TCP/IP input and output

NIC ring

/mnt/nvmm/myapp_metadata

/mnt/nvmm/pktbufs

In Action



Unread

Read or written

Flushed

Application

(1) Read data (zero copy)

(2) Write metadata entry

(3) Flush (buffer and metadata)

User

Kernel

mmap()

netmap API

metadata header

/mnt/nvmm/pktbufs

buf_ofs: 123

metadata entries

buf_idx	off	len
0	100	1135
1	100	932
3	100	1024

packet buffers (static)

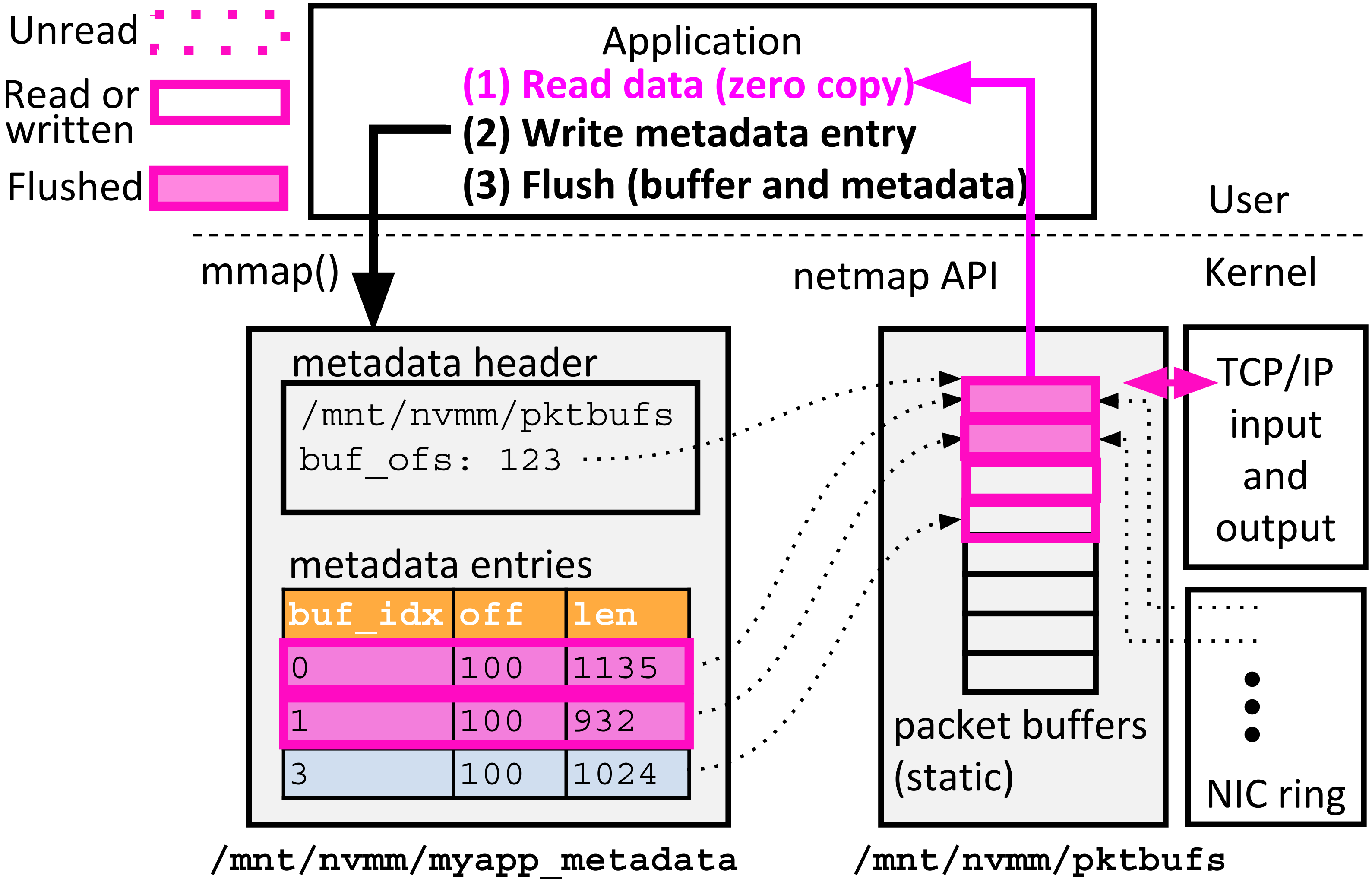
TCP/IP input and output

NIC ring

/mnt/nvmm/myapp_metadata

/mnt/nvmm/pktbufs

In Action



Unread

Read or written

Flushed

Application

(1) Read data (zero copy)

(2) Write metadata entry

(3) Flush (buffer and metadata)

User

Kernel

mmap()

netmap API

metadata header

```
/mnt/nvmm/pktbufs
buf_ofs: 123
```

metadata entries

buf_idx	off	len
0	100	1135
1	100	932
3	100	1024

packet buffers (static)

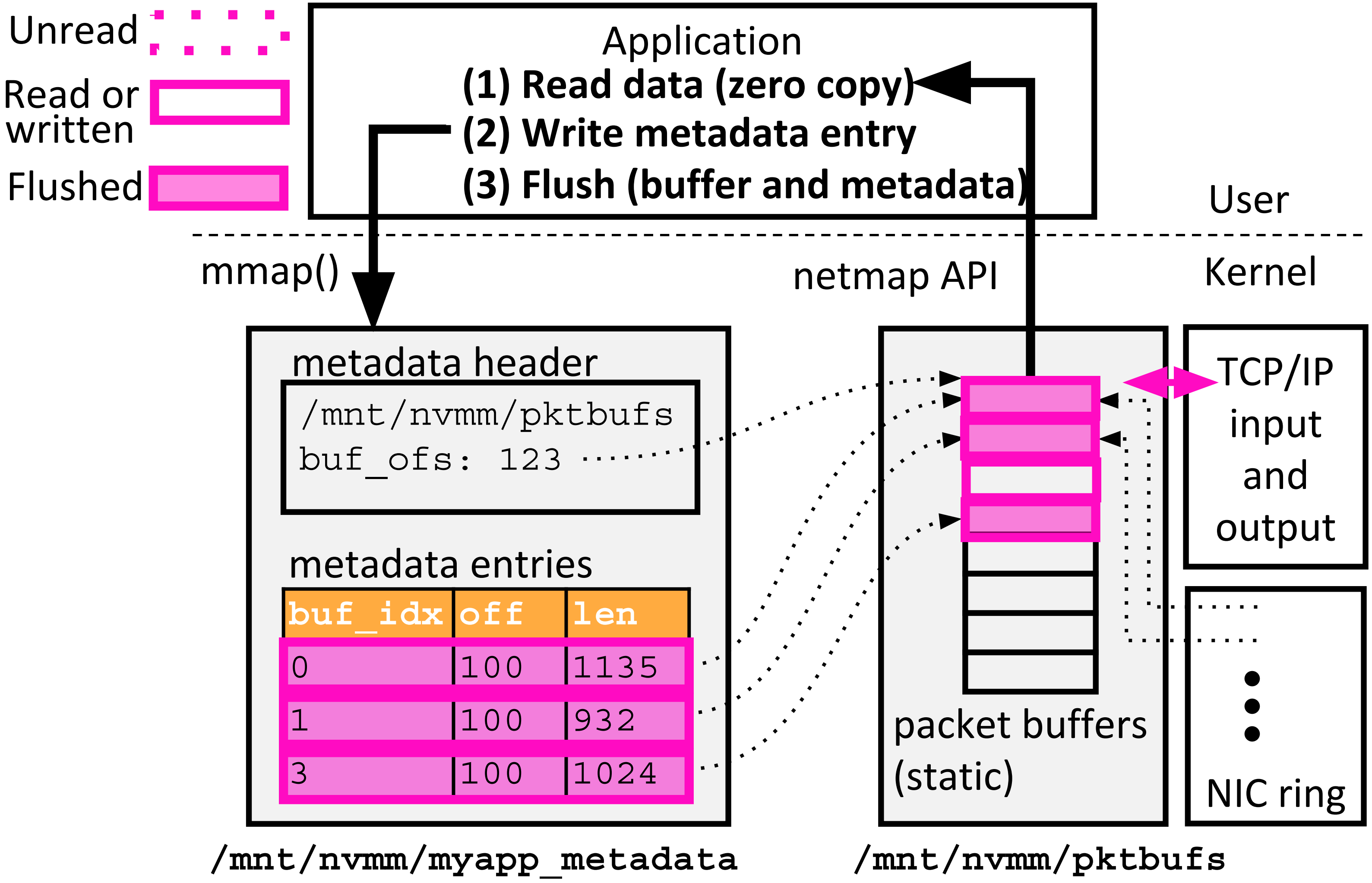
TCP/IP input and output

NIC ring

/mnt/nvmm/myapp_metadata

/mnt/nvmm/pktbufs

In Action



Application

(1) Read data (zero copy)

(2) Write metadata entry

(3) Flush (buffer and metadata)

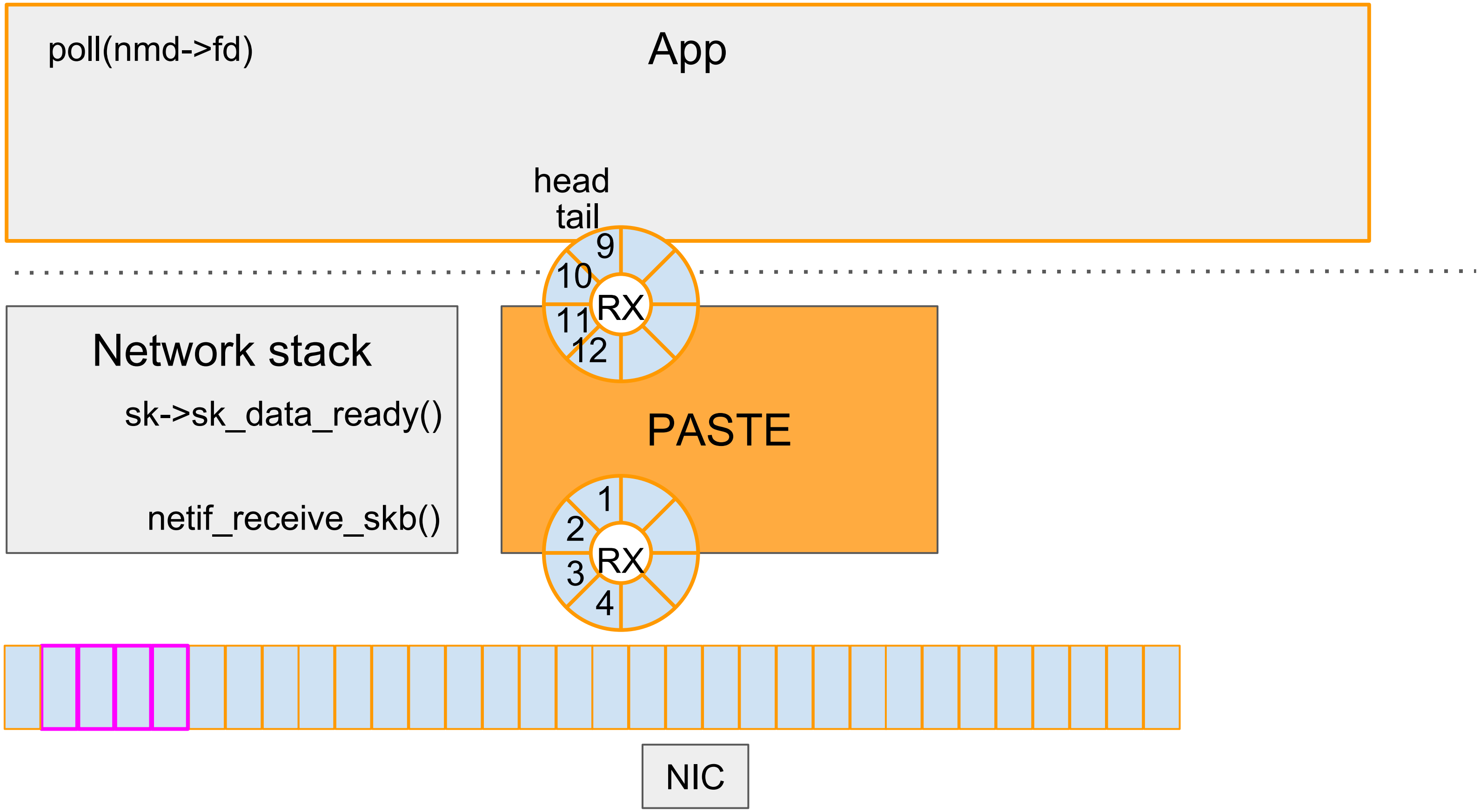
metadata header

/mnt/nvmm/pktbufs
buf_ofs: 123

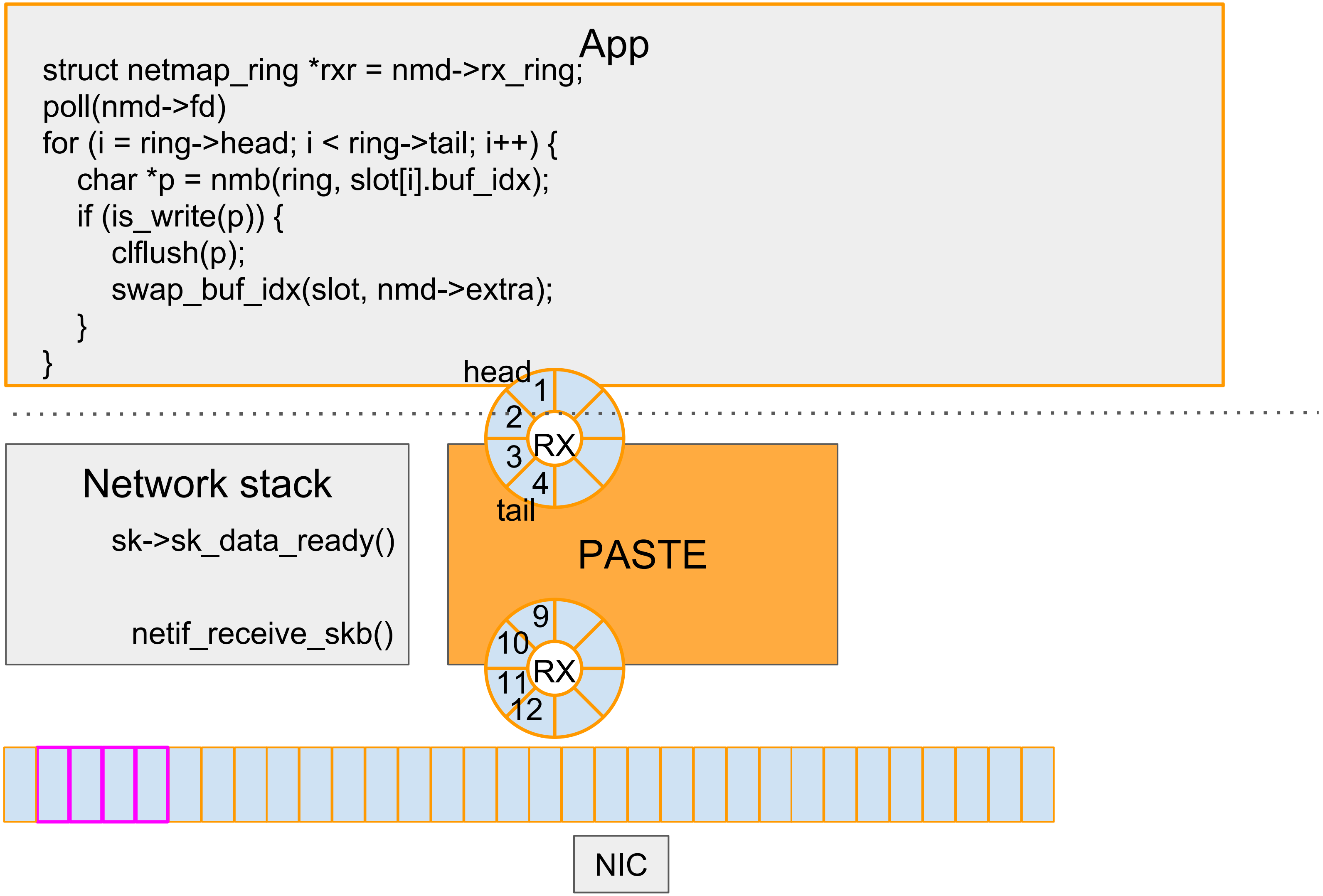
metadata entries

buf_idx	off	len
0	100	1135
1	100	932
3	100	1024

In Action


poll(nmd->fd)
App
head
tail
9
10
11
12
RX
Network stack
sk->sk_data_ready()
netif_receive_skb()
PASTE
1
2
3
4
RX
NIC

In Action



```
struct netmap_ring *rxr = nmd->rx_ring;
poll(nmd->fd)
for (i = ring->head; i < ring->tail; i++) {
    char *p = nmb(ring, slot[i].buf_idx);
    if (is_write(p)) {
        clflush(p);
        swap_buf_idx(slot, nmd->extra);
    }
}
```

Challenges

- Moving packets to and from the TCP/IP stack
  - Two-level skb destructors
  - sk’s callbacks
- Avoiding HoL
  - Buffer swapping to drain sender or NIC queue
- Getting kernel pages
  - get_user_pages()

Performance

Conclusion