

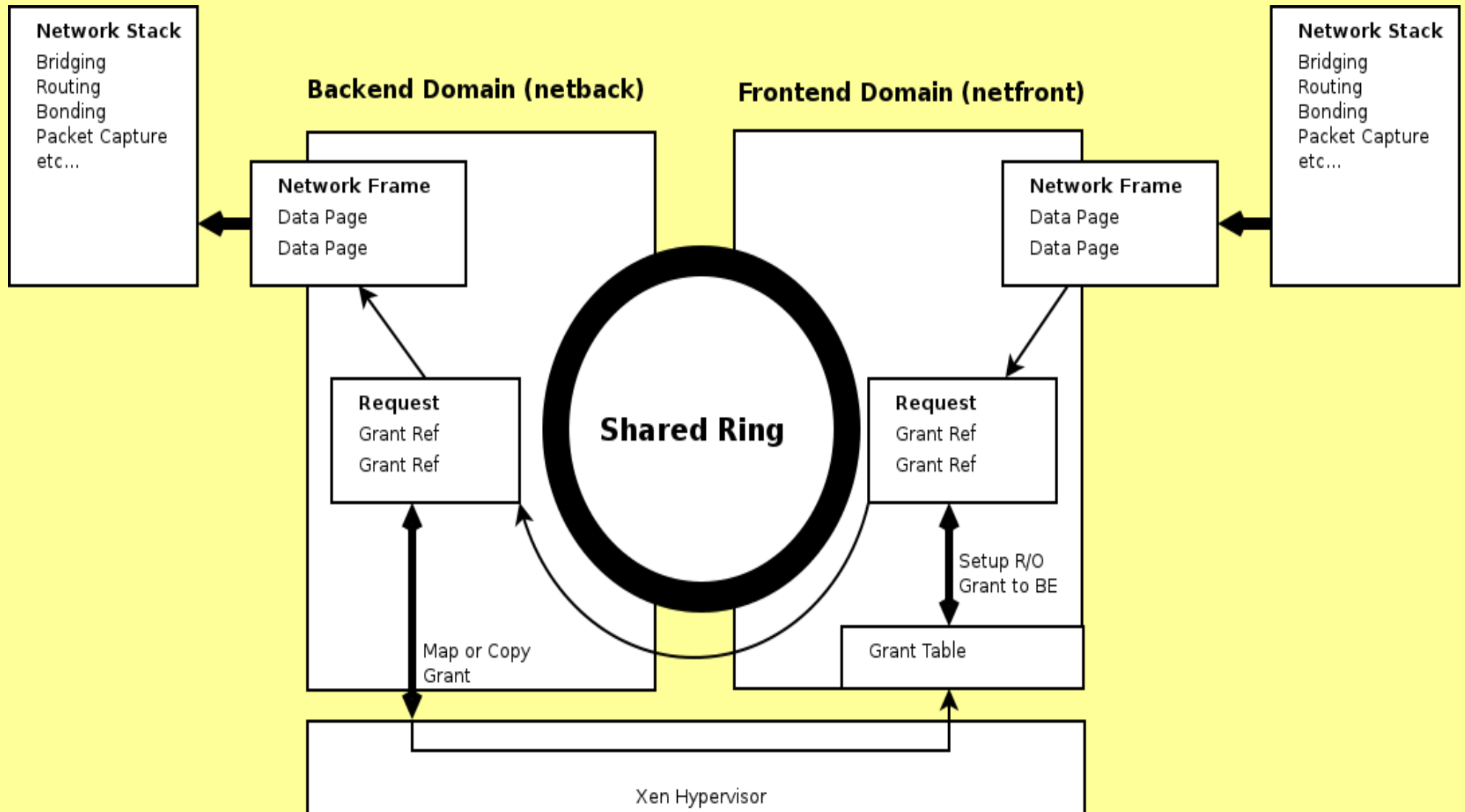
Zero-Copy Networking (or SKB fragment life-cycle tracking)

Ian Campbell
Citrix & Xen.org
<ian.campbell@citrix.com>

Agenda

- Xen PV network transmit operation
- Issues with supporting zero-copy
- Similar issues seen elsewhere
- How can we solve these problems
- SKB paged fragment destructors

Xen PV Network Transmit



Problems With Mapping

- Must eventually unmap and return to guest
- Must give up complete control of page to network stack
- Cannot do both...
 - “Foreign” memory free()d into standard allocation pools
 - Cannot manipulate mappings of foreign pages in the normal way (set_pte etc)
 - Badness occurs!

Underlying Issue

- Network stack does no tracking of page references itself
- Piggybacks on the core's reference counting of struct page
 - Uses raw `get_page()` and `put_page()`

“Classic” Xen Patch Solution

- `include/linux/page-flags.h:`

```
#ifdef CONFIG_XEN
    PG_foreign, /* Page is owned by foreign allocator. */
#endif

...
static inline void SetPageForeign(struct page *page,
    void (*dtor)(struct page *, unsigned int)) {
    set_bit(PG_foreign, &page->flags);
    page->index = (long)dtor;
}

static inline void PageForeignDestructor(struct page *page,...) {
    ((void (*)(struct page *, unsigned int))page->index)(page,...);
}
```

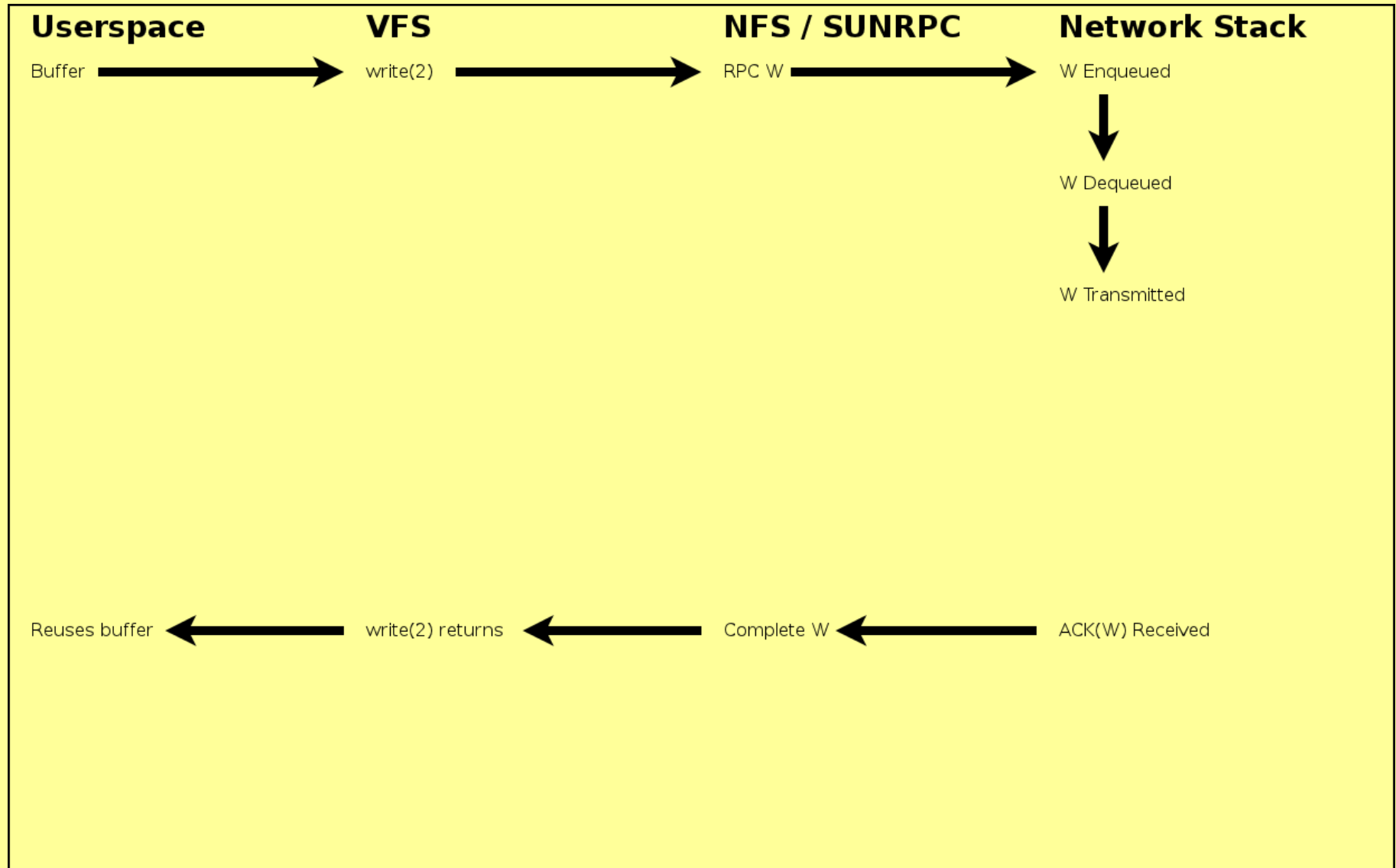
- `mm/page_alloc.c:`

```
static void free_hot_cold_page(struct page *page, int cold) {
    ...
#ifdef CONFIG_XEN
    if (PageForeign(page)) {
        PageForeignDestructor(page, 0);
        return;
    }
}
```

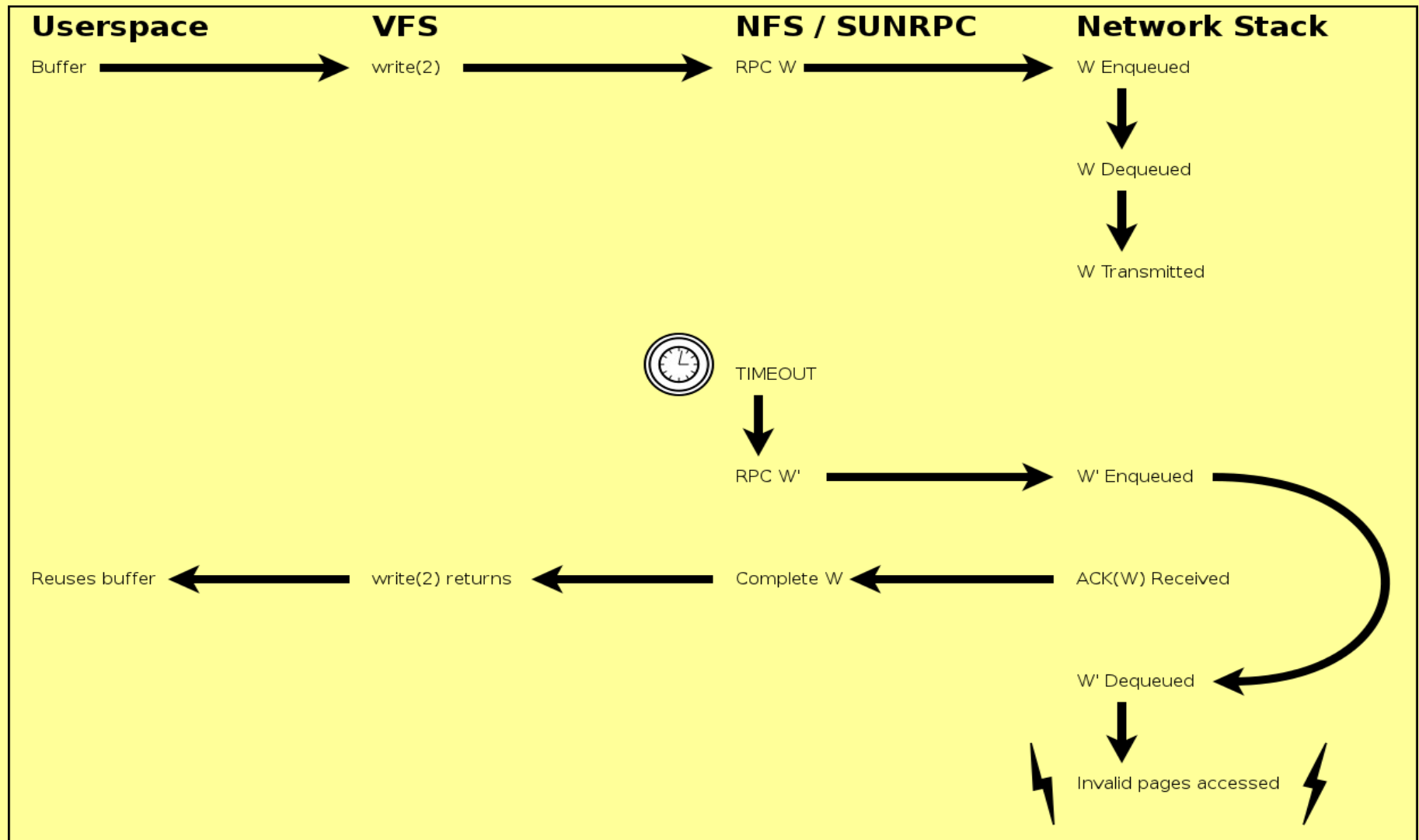
Netback in mainline

- PageForeign approach unsuitable for mainline
- netback upstreamed with copy instead of map
- Better than nothing...
- ... but obviously we would like zero-copy again

NFS Write



NFS Write Retransmit



Reproduction

- `iptables -I INPUT --source $SERVER -j QUEUE`
- `netoutage.c`:
 - delay $1/N$ packets for 60s using `libnetfilter_queue`
- `test.c`:

```
{  
    ...  
    memset(buf, SIZE, 0xAA);  
    write(fd, buf, SIZE);  
    memset(buf, SIZE, 0x55);  
    ...  
}
```
- `tcpdump... 'host $SERVER and ip[184:4] == 0x55555555'`

Captured Result

```

> Frame 2466: 1514 bytes on wire (12112 bits), 1514 bytes captured (12112 bits)
> Ethernet II, Src: Dell_75:1f:bf (00:24:e8:75:1f:bf), Dst: Intel_2a:09:2e (00:19:d1:2a:09:2e)
> Internet Protocol Version 4, Src: 10.80.227.104 (10.80.227.104), Dst: 10.80.224.22 (10.80.224.22)
> Transmission Control Protocol, Src Port: 870 (870), Dst Port: nfs (2049), Seq: 949617051, Ack: 1744437828, Len: 1448
▼ Remote Procedure Call
  Continuation data

```

```

0000 00 19 d1 2a 09 2e 00 24 e8 75 1f bf 08 00 45 00 ...*...
0010 05 dc 4d 54 40 00 40 06 0f a9 0a 50 e3 68 0a 50 ..MT@.
0020 e0 16 03 66 08 01 38 9a 01 9b 67 fa 02 44 80 10 ...f..
0030 01 f5 dd ed 00 00 01 01 08 0a 00 02 db 5c 03 e6 .....
0040 a2 fc aa aa aa aa aa aa aa aa aa aa aa aa aa aa .....
0050 aa aa aa aa aa aa aa aa aa aa aa aa aa aa aa .....
0060 aa aa aa aa aa aa aa aa aa aa aa aa aa aa aa .....
0070 aa aa aa aa aa aa aa aa aa aa aa aa aa aa aa .....
0080 aa aa aa aa aa aa aa aa aa aa aa aa aa aa aa .....
0090 aa aa aa aa aa aa aa aa aa aa aa aa aa aa aa .....
00a0 aa aa aa aa aa aa aa aa aa aa aa aa aa aa aa .....
00b0 aa aa aa aa aa aa aa aa aa aa aa aa aa aa aa .....
00c0 aa aa aa aa aa aa aa aa aa aa aa aa aa aa aa .....
00d0 aa aa aa aa aa aa aa aa aa aa aa aa aa aa aa .....
00e0 aa aa aa aa aa aa aa aa aa aa aa aa aa aa aa .....
00f0 aa aa aa aa aa aa aa aa aa aa aa aa aa aa aa .....
0100 aa aa aa aa aa aa aa aa aa aa aa aa aa aa aa .....
0110 aa aa aa aa aa aa aa aa aa aa aa aa aa aa aa .....
0120 aa aa aa aa aa aa aa aa aa aa aa aa aa aa aa .....
0130 aa aa aa aa aa aa aa aa aa aa aa aa aa aa aa .....
0140 aa aa aa aa aa aa aa aa aa aa aa aa aa aa aa .....
0150 aa aa aa aa aa aa aa aa aa aa aa aa aa aa aa .....
0160 aa aa aa aa aa aa aa aa aa aa aa aa aa aa aa .....
0170 aa aa aa aa aa aa aa aa aa aa aa aa aa aa aa .....
0180 aa aa aa aa aa aa aa aa aa aa aa aa aa aa aa .....
0190 aa aa aa aa aa aa aa aa aa aa aa aa aa aa aa .....
01a0 aa aa aa aa aa aa aa aa aa aa aa aa aa aa aa .....
01b0 aa aa aa aa aa aa aa aa aa aa aa aa aa aa aa .....
01c0 aa aa aa aa aa aa aa aa aa aa aa aa aa aa aa .....
01d0 aa aa aa aa aa aa aa aa aa aa aa aa aa aa aa .....
01e0 aa aa aa aa aa aa aa aa aa aa aa aa aa aa aa .....
01f0 aa aa aa aa aa aa aa aa aa aa aa aa aa aa aa .....
0200 aa aa aa aa aa aa aa aa aa aa aa aa aa aa aa .....
0210 aa aa aa aa aa aa aa aa aa aa aa aa aa aa aa .....

```

```

> Frame 2507: 2962 bytes on wire (23696 bits), 2962 bytes captured (23696 bits)
> Ethernet II, Src: Dell_75:1f:bf (00:24:e8:75:1f:bf), Dst: Intel_2a:09:2e (00:19:d1:2a:09:2e)
> Internet Protocol Version 4, Src: 10.80.227.104 (10.80.227.104), Dst: 10.80.224.22 (10.80.224.22)
> Transmission Control Protocol, Src Port: 870 (870), Dst Port: nfs (2049), Seq: 949622727, Ack: 1744437968, Len: 2896
▼ Remote Procedure Call
  Continuation data

```

```

0000 00 19 d1 2a 09 2e 00 24 e8 75 1f bf 08 00 45 00 ...*...$ .u...E.
0010 0b 84 4d 68 40 00 40 06 09 ed 0a 50 e3 68 0a 50 ..Mh@.@. ...P.h.P
0020 e0 16 03 66 08 01 38 9a 17 c7 67 fa 02 d0 80 10 ...f..8. ..g....
0030 01 f5 e3 95 00 00 01 01 08 0a 00 02 f4 c1 03 e6 .....
0040 e2 73 55 55 55 55 55 55 55 55 55 55 55 55 55 ..sUUUUUU UUUUUUU
0050 55 55 55 55 55 55 55 55 55 55 55 55 55 55 55 UUUUUUUU UUUUUUUU
0060 55 55 55 55 55 55 55 55 55 55 55 55 55 55 55 UUUUUUUU UUUUUUUU
0070 55 55 55 55 55 55 55 55 55 55 55 55 55 55 55 UUUUUUUU UUUUUUUU
0080 55 55 55 55 55 55 55 55 55 55 55 55 55 55 55 UUUUUUUU UUUUUUUU
0090 55 55 55 55 55 55 55 55 55 55 55 55 55 55 55 UUUUUUUU UUUUUUUU
00a0 55 55 55 55 55 55 55 55 55 55 55 55 55 55 55 UUUUUUUU UUUUUUUU
00b0 55 55 55 55 55 55 55 55 55 55 55 55 55 55 55 UUUUUUUU UUUUUUUU
00c0 55 55 55 55 55 55 55 55 55 55 55 55 55 55 55 UUUUUUUU UUUUUUUU
00d0 55 55 55 55 55 55 55 55 55 55 55 55 55 55 55 UUUUUUUU UUUUUUUU
00e0 55 55 55 55 55 55 55 55 55 55 55 55 55 55 55 UUUUUUUU UUUUUUUU
00f0 55 55 55 55 55 55 55 55 55 55 55 55 55 55 55 UUUUUUUU UUUUUUUU
0100 55 55 55 55 55 55 55 55 55 55 55 55 55 55 55 UUUUUUUU UUUUUUUU
0110 55 55 55 55 55 55 55 55 55 55 55 55 55 55 55 UUUUUUUU UUUUUUUU
0120 55 55 55 55 55 55 55 55 55 55 55 55 55 55 55 UUUUUUUU UUUUUUUU
0130 55 55 55 55 55 55 55 55 55 55 55 55 55 55 55 UUUUUUUU UUUUUUUU
0140 55 55 55 55 55 55 55 55 55 55 55 55 55 55 55 UUUUUUUU UUUUUUUU
0150 55 55 55 55 55 55 55 55 55 55 55 55 55 55 55 UUUUUUUU UUUUUUUU
0160 55 55 55 55 55 55 55 55 55 55 55 55 55 55 55 UUUUUUUU UUUUUUUU
0170 55 55 55 55 55 55 55 55 55 55 55 55 55 55 55 UUUUUUUU UUUUUUUU
0180 55 55 55 55 55 55 55 55 55 55 55 55 55 55 55 UUUUUUUU UUUUUUUU
0190 55 55 55 55 55 55 55 55 55 55 55 55 55 55 55 UUUUUUUU UUUUUUUU
01a0 55 55 55 55 55 55 55 55 55 55 55 55 55 55 55 UUUUUUUU UUUUUUUU
01b0 55 55 55 55 55 55 55 55 55 55 55 55 55 55 55 UUUUUUUU UUUUUUUU
01c0 55 55 55 55 55 55 55 55 55 55 55 55 55 55 55 UUUUUUUU UUUUUUUU
01d0 55 55 55 55 55 55 55 55 55 55 55 55 55 55 55 UUUUUUUU UUUUUUUU
01e0 55 55 55 55 55 55 55 55 55 55 55 55 55 55 55 UUUUUUUU UUUUUUUU
01f0 55 55 55 55 55 55 55 55 55 55 55 55 55 55 55 UUUUUUUU UUUUUUUU
0200 55 55 55 55 55 55 55 55 55 55 55 55 55 55 55 UUUUUUUU UUUUUUUU
0210 55 55 55 55 55 55 55 55 55 55 55 55 55 55 55 UUUUUUUU UUUUUUUU

```

Requirements for a General Solution

- Callers to be able to give a page to the network stack while retaining ownership
- Not sufficient to just know when page is freed
 - There may be other holders of core kernel references on a page
- Need to track all references with the stack across `skb_clone`, `skb_pull_up`, segmentation etc
 - Existing `skb` destructor not sufficient

SKB Paged Fragment Usage

```
struct skb_frag_struct {  
    struct page *page;  
    ...  
};  
...  
frag->page = page;  
...  
get_page(skb_shinfo(skb)->frags[i].page)  
...  
dma_map_page(..., frag->page, ...)  
...  
put_page(skb_shinfo(skb)->frags[i].page)
```


SKB Paged Fragment API

```
static inline struct page *skb_frag_page(const skb_frag_t *frag)
...
static inline void skb_frag_ref(struct sk_buff *skb, int f)
...
static inline void skb_frag_unref(struct sk_buff *skb, int f)
...
static inline dma_addr_t skb_frag_dma_map(
    struct device *dev,
    const skb_frag_t *frag,
    size_t offset, size_t size,
    enum dma_data_direction dir)
...
```

SKB Paged Fragment API Patches

- Touches a lot of code
87 files changed, 513 insertions(+), 356 deletions(-)
- In particular touches most network drivers
- Coccinelle (spatch) was very useful in the early stages of the conversion
- API itself is in net-next
- Driver conversions are under way as well
 - ~ half complete.

SKB Paged Fragment Destructors

```
struct skb_frag_destructor {  
    atomic_t ref;  
    int (*destroy)(void *data);  
    void *data;  
};  
  
struct skb_frag_struct {  
    struct {  
        struct page *p;  
        struct skb_frag_destructor *destr...;  
    } page;  
    ...  
};
```

SKB Paged Fragment Destructors

- Builds on the Fragment API, will come after
- Far less intrusive

```
include/linux/skbuff.h | 54 ++++++[...]++++++-----  
net/core/skbuff.c      | 26 ++++++-----  
2 files changed, 74 insertions(+), 6 deletions(-)
```

- Additional patch uses it in the RPC layer

```
include/linux/sunrpc/xdr.h | 2 ++  
include/linux/sunrpc/xprt.h | 5 +++-  
net/sunrpc/clnt.c          | 27 ++++++-----  
net/sunrpc/svcsock.c       | 3 +-  
net/sunrpc/xprt.c          | 13 ++++++  
net/sunrpc/xprtsock.c      | 3 +-  
6 files changed, 45 insertions(+), 8 deletions(-)
```


Summary

- Fragment API trickling into net-next
- Fragment destructor patches ready
- NFS / SUNRPC fix ready
- Next steps:
 - Zero-copy in netback
 - iSCSI?
 - Packet mmap?



Questions?