

Zero-Copy BPF



Robert N. M. Watson
Computer Laboratory
University of Cambridge

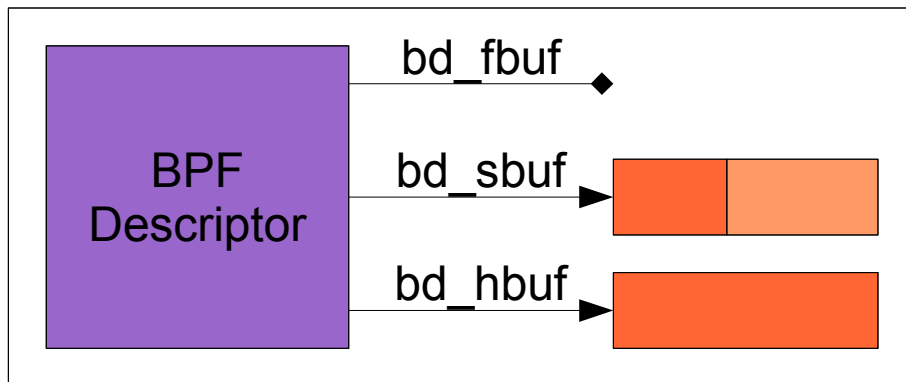
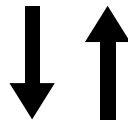
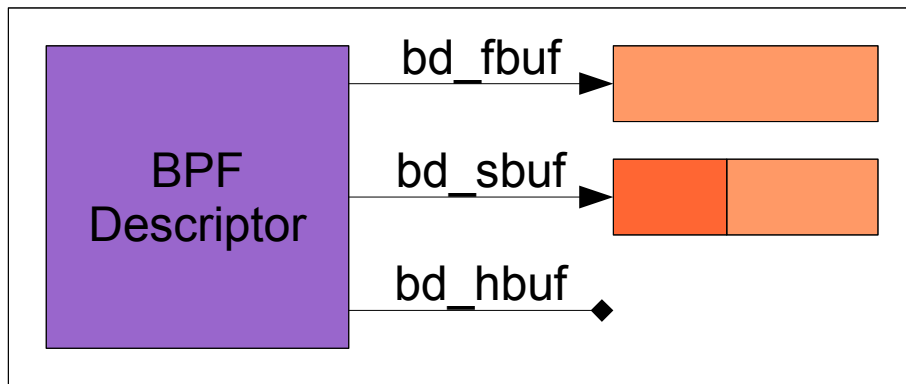
Christian S. J. Peron
Seccuris Inc.

9 March, 2007

BPF: Berkeley Packet Filter

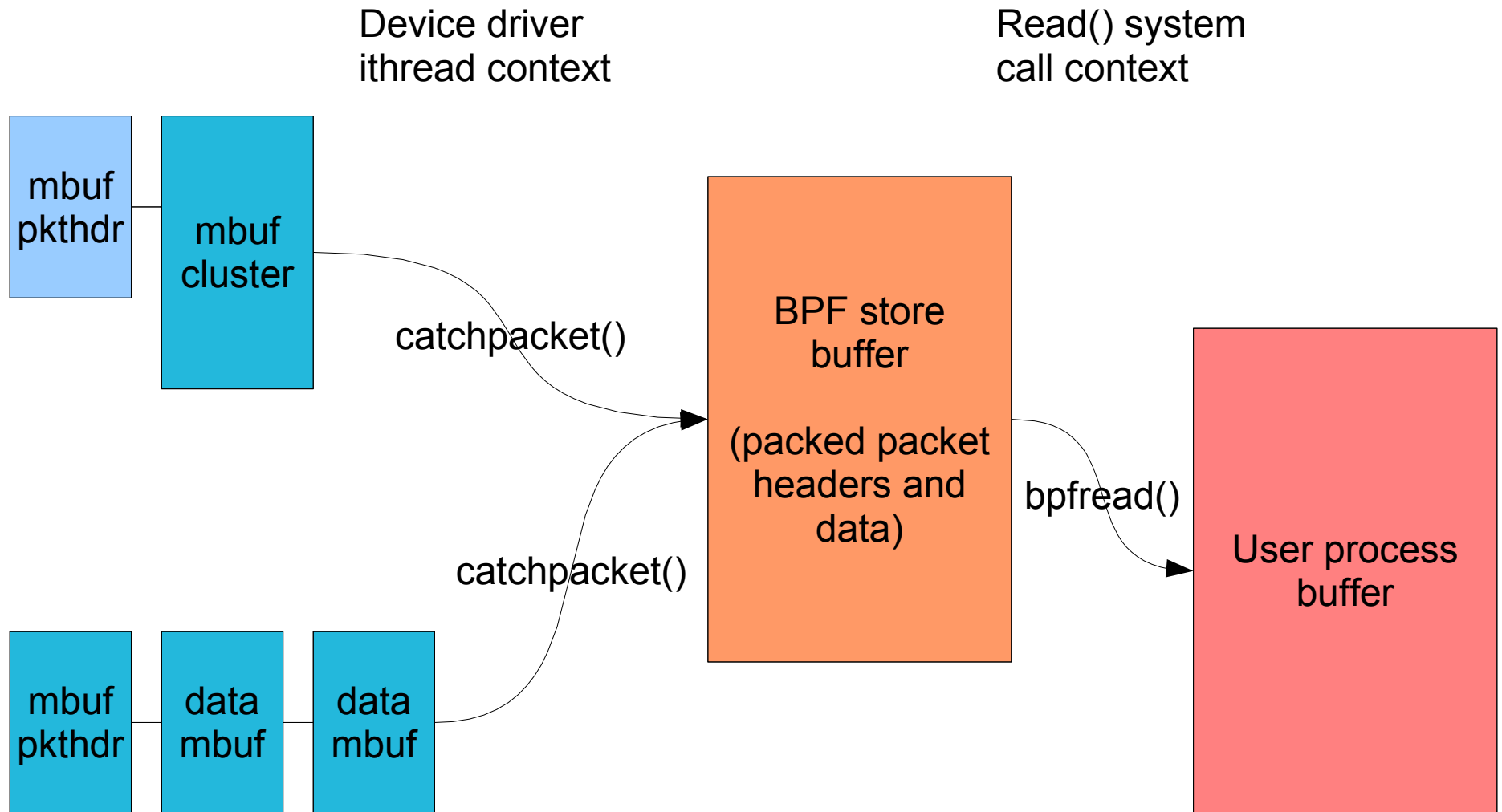
- BPF provides user process interface for link layer receive and transmit
 - Open special device `/dev/bpfX`
 - Program in-kernel packet filter
 - Select interface, optionally set promiscuous mode
 - Loop on `read()` to read buffers of a fixed size
- Problem: minimum of two copies per packet
 - mbufs -> kernel buffer -> user memory
 - A significant performance overhead

BPF Buffer Model



- Two rotating buffers per descriptor
- Rotate between free, store, and hold buffers
- Hold buffer returns to free slot after `bpfread()` drains to user space

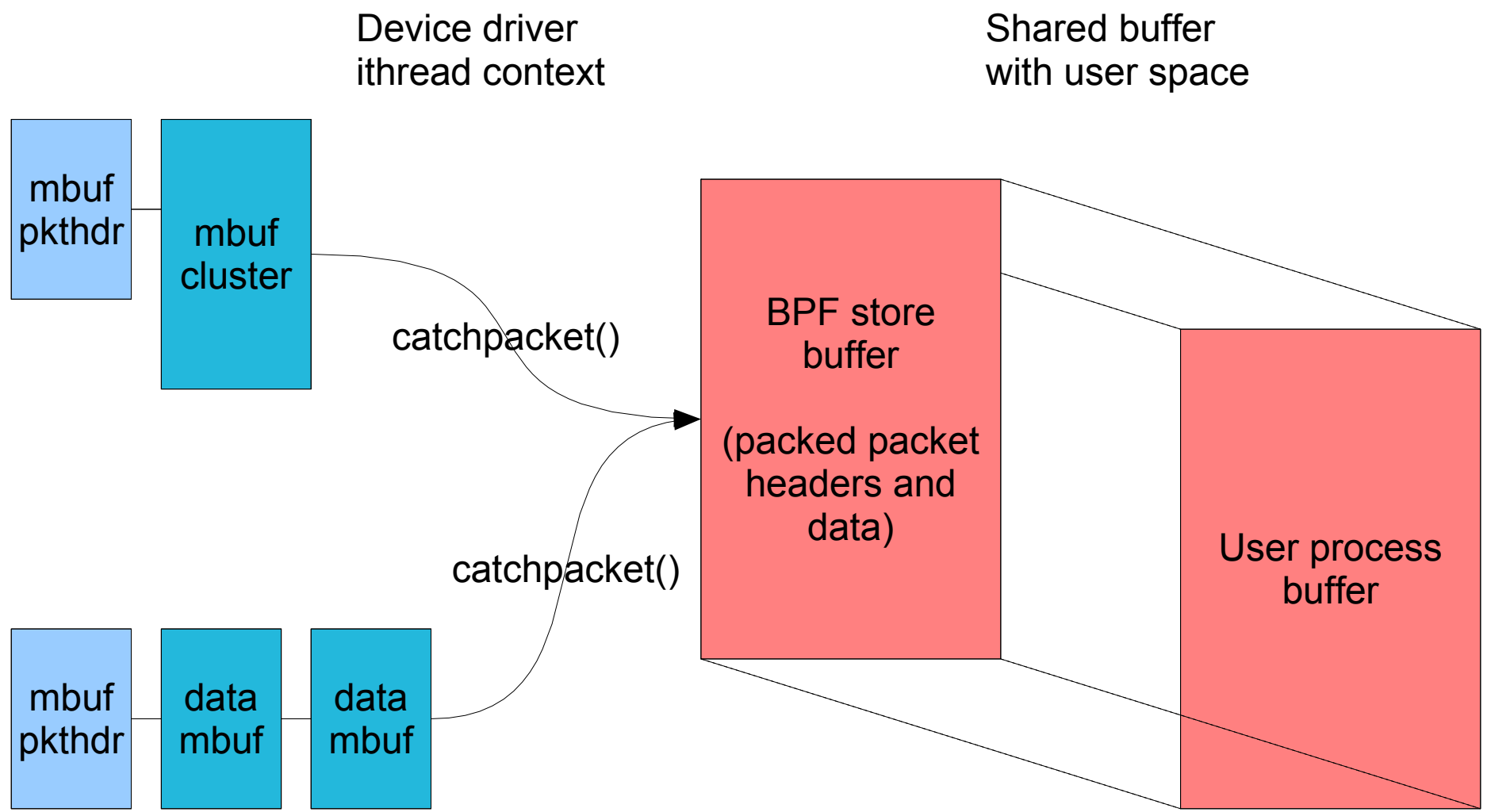
BPF Data Copies (Currently)



Zero-Copy BPF

- Problem: too many data copies
- Solution: shared memory buffer between user process and BPF
- Eliminates copy to user space, not in-kernel
 - Strictly, now “one-copy”
 - In-kernel copy prevents leaking of kernel memory
 - Allows independence between BPF devices (different filters, snaplens, etc)

Shared Memory BPF Store Buffer



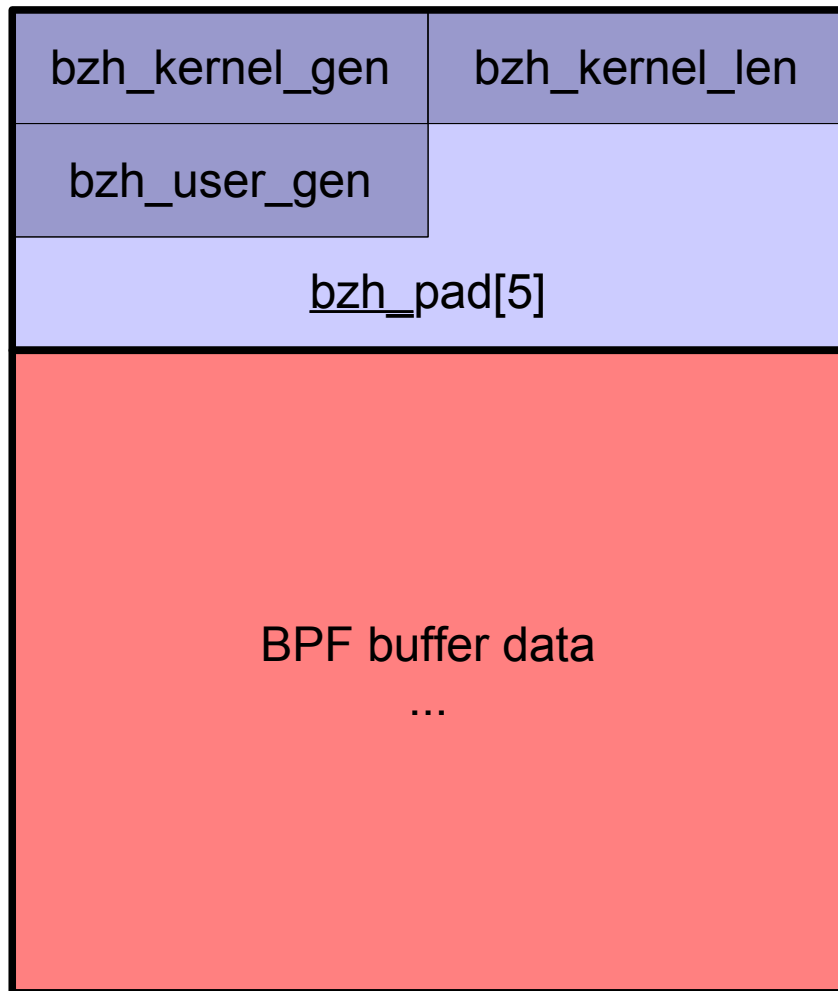
Shared BPF Buffers

- User process
 - Selects non-default mode `BPF_BUFFER_ZBUF`
 - Allocates two page-aligned, identically sized buffers
 - Set buffer size and locations with `BIOCSETZBUF`
- Kernel
 - Maps user buffers into kernel address space
 - Pins into physical memory
 - Uses buffers instead of kernel memory for `bd_{fbuf, sbuf, hbuf}`

Shared BPF Buffers (cont)

- User process may use syscalls and ioctls:
 - Poll(), etc, to monitor for complete buffers
 - Query next held buffer using BIOCGETZNEXT
 - Force rotation using BIOCROTZBUF to time out partially filled buffer
 - Return held buffer to free slot using BIOACKZBUF
- Or query and acknowledge buffers using a shared memory header at the front of the buffer
 - Forced rotation still requires BIOCROTZBUF

Memory-Mapped BPF Buffer Layout



- Memory buffer prefixed with shared memory header
 - Used for system call free synchronization between kernel and user app
- Remainder of buffer uses standard BPF buffer layout

BPF Implementation Changes

- Abstract buffer access
 - Default is BPF_BUFMODE_BUFFER uses kernel memory and full data copies
 - Optional BPF_BUFMODE_ZBUF uses shared user/kernel memory buffers with reduced copies
- New ioctls
 - Configure, manage shared memory buffers
- Libpcap
 - Modified to speak both models

Current Status

- Not much testing or evaluation yet, but works
- Faster in some benchmarks, slower in others
 - Wins for simple buffer traversal benchmark...
 - ... but not for complex memory scanning benchmark
 - Could be experimental error (not well-tested yet)
 - Could be increased overhead of scatter-gather copy?
 - Cache misses also moved around, may need work
 - Or might just not be faster not to copy
- Need to do a full hwpmc analysis, comprehensive benchmarking

Where to get it?

- Perforce: `//depot/projects/zcopybpf/...`
- Coordinate with Christian Peron (csjp@) and Robert Watson (rwatson@)
- Sponsored by Seccuris Inc