

SeaOS: A simple OS for multicore machines

or: how I learned to stop worrying and love mmap

Daniel Bittman
SSRC - UC Santa Cruz - 2016

2016 USENIX Vail Computer Elements Workshop

Okay, I'm going to try something here...

SeaOS is both old, and new.

- I've been writing kernel code for a while.
- This particular iteration is only a few months old.

Why?

- Learning experience!
- Research
 - Demonstrated a VM monitor breakout in 2015 [1]

SeaOS is for *modern systems*

Requires 64-bit, MMU systems

Supports SMP

Runs on x86_64, armv8 (aarch64)

SeaOS is *simple*

Kernel	Lines of Code
Linux	~1,000,000
SeaOS	4,500

(this excludes drivers and fs code)

Generated using David A. Wheeler's 'SLOCCount'

SeaOS is *portable*

- 1800 lines of x86_64 specific code
- 1150 lines of armv8 specific code


(some) Design considerations

- *At this point*, assuming SMP
- Core kernel has minimal global shared state
- Multikernel considerations? [2]
- Can I make threading more *low overhead*?

Unix compatibility layer

Currently implemented in-kernel.

Would like to move as much as possible to userspace...



My favorite syscall:
mmap

mmap is really cool

SeaOS has mmap (and friends)

The normal stuff works:

- Userspace can call mmap
- ELF files are loaded by mmap-ing program headers (SeaOS has dynamic linking, too)
- Can map other things, like devices (/dev/vga).

SeaOS needs a networking stack

Could implement one in-kernel...

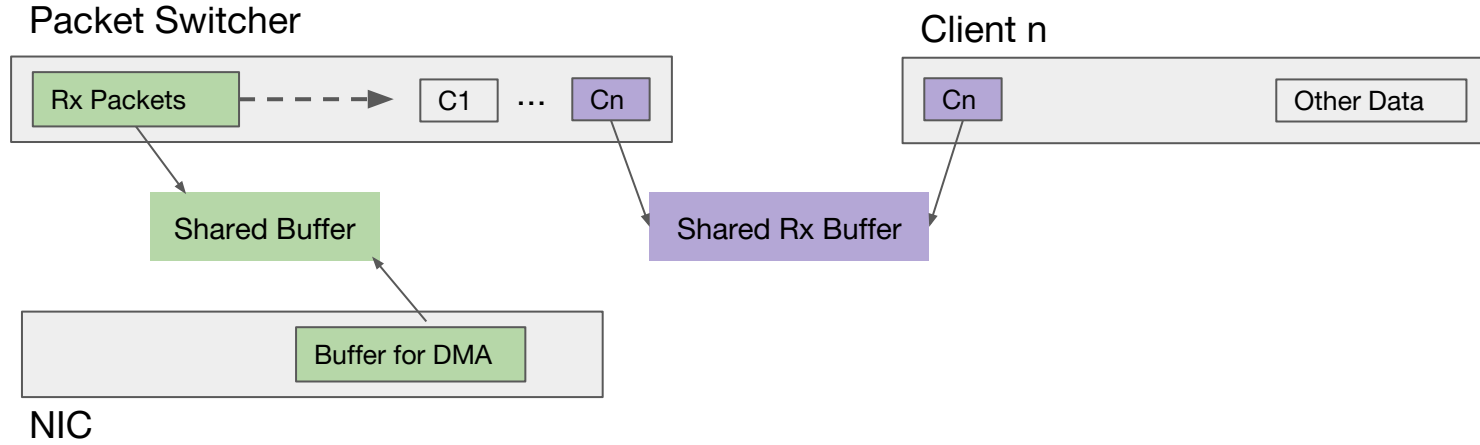
...but there's mmap!

SeaOS needs a networking stack

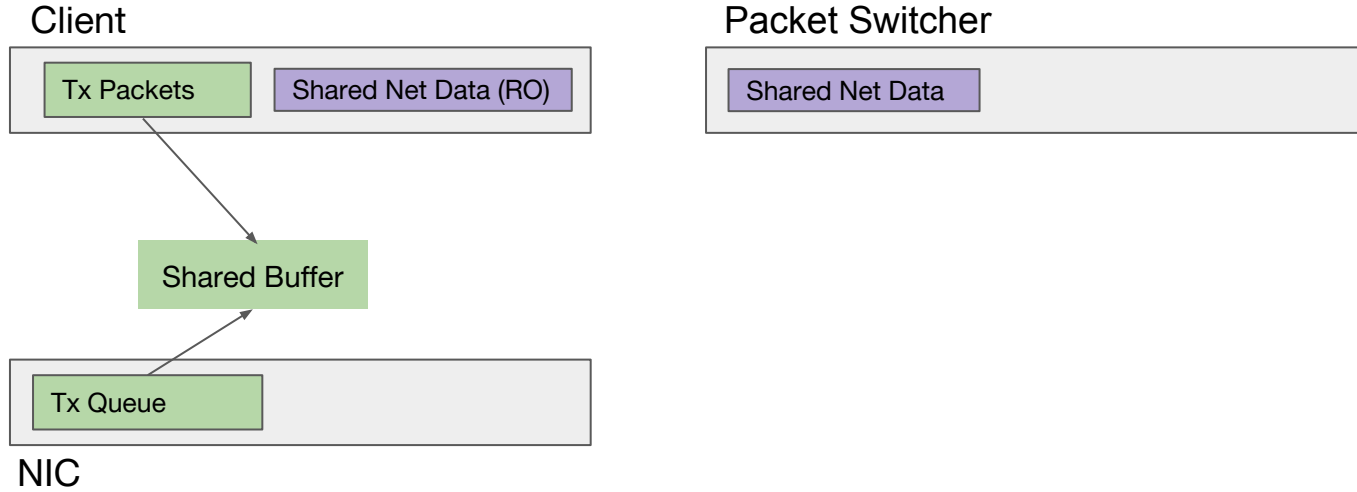
Most obvious implementation

- “Packet Switcher” process shares memory with the NIC [3]
- NIC reads packets from the networking into that memory
- ...the packets are now in userspace!

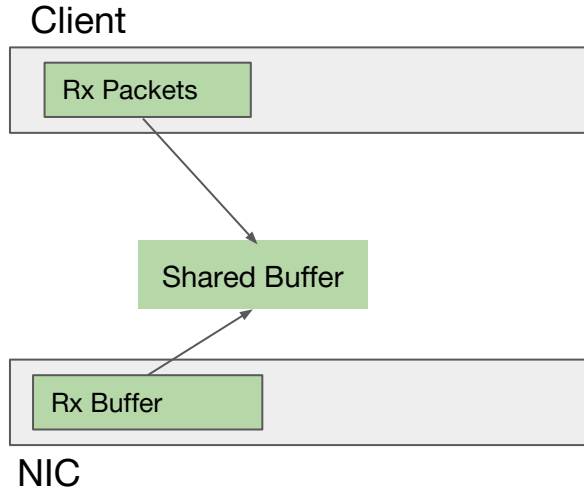
Receiving



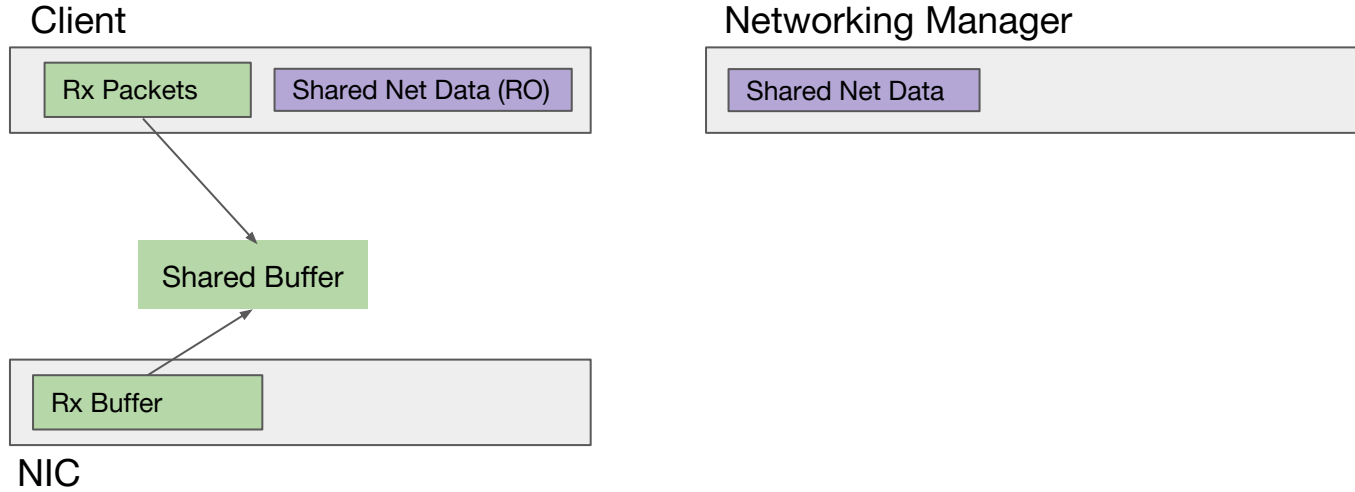
Transmitting



Zero-copy: even more sharing



Zero-copy: *even more* sharing



Other UNIX System Calls?

SeaOS has read, readv, write, etc.

...can they be implemented with mmap?

Other UNIX System Calls?

- read is easy, especially if the buffer is page-aligned.
- readv is harder (atomicity on some systems)
- write is harder too
- FS manipulation syscalls?

Concurrency



Simplicity helps with reasoning about the complexities of a highly concurrent system

Assuming SMP, but allow for multikernel-like behavior

Investigate lock-free and low-lock data structures and algorithms

Conclusion

SeaOS is a simple, portable, and easy-to-modify OS.

I plan to use SeaOS for explorations of interesting academic concepts.

Thank You

Daniel Bittman

danielbittman1@gmail.com

dbittman.github.io

@danielbittman

This talk is available on github:
github.com/dbittman/talk-mmap-everywhere

References

1. DJ Capelis, Daniel Bittman. "The Hypervisor Exploit I Sat on for Five Years". PoC||GTFO 0x08, June, 2015.
2. Baumann, Andrew, et al. "The multikernel: a new OS architecture for scalable multicore systems." Proceedings of the ACM SIGOPS 22nd symposium on Operating systems principles. ACM, 2009.
3. Rizzo, Luigi. "Netmap: a novel framework for fast packet I/O." 21st USENIX Security Symposium (USENIX Security 12). 2012.