# Porting Kernel Code to Four BSDs and Linux

Craig Metz

June 10, 1999

# Overview

- Introduction/Background
- Should You Port It?
- General Techniques
- The `nbuf`
- Conclusions

# Introduction

- This talk is based on my group's experience with the NRL IPv6+IPsec code.
- IPv6 code runs on BSD/OS, FreeBSD, NetBSD, OpenBSD.
- IPsec code runs on those plus Linux (port in progress).

# Introduction

- Porting kernel code is not a new idea.
- `de` driver $\rightarrow$ all four BSDs.
- Linux x87 emulator $\rightarrow$ *BSD
- BSD 53c*xxx* driver $\rightarrow$ Linux
- BSD network code $\rightarrow$ sundry

# Introduction

- Yet, there's a big resistance to porting kernel code.
- An overemphasis on the differences between systems.
- Standards put constraints on the differences.

# Should You Port It?

- Be realistic. You can't port everything.
- Some things depend too heavily on the original system.
- It would be less effort to just rewrite those.

# Should You Port It?

- Some problems on one system don't exist on another.
- Or are being solved by somebody else.
- Pick problems that need you to solve them.

# Should You Port It?

- Making useful features available on more systems is a good thing.
- Making systems do the same thing similarly is good.
- You don't have to have 99% code sharing to be useful.

# Techniques

- Use what you (hopefully) already know from userland.
- But bad code is tolerated less by kernel people.
- I'll mention a few points here.
- See the paper for more.

# Techniques

- Organize code carefully.
- You'll go crazy if you don't.
- Our tree has seven pieces:
  - One for each system (x5)
  - One shared by all BSD
  - One shared by all

# Techniques

- Abstraction: Macros, higher-level functions, etc.
- Conditionalizing: A maze of twisty `ifdef` statements
- Prefer the former, but prefer doesn't mean always

# Techniques

- Linux: `kmalloc(size, flags)`
- BSD: `malloc(size, type, wait)`
- Abstraction: `OSDEP_MALLOC(size)`
- For similar things, trade off unneeded features for generality

# Techniques

- Linux: `__u32`
- BSD: `u_int32_t`
- POSIX: `uint32_t`
- Same thing, different names.
- We make the POSIX names available in all systems and use those.

# The `nbuf`

- Linux: `struct sk_buff`
- BSD: `struct mbuf`
- No `#define` for this one.
- Deeper differences require more work to handle.
- We created an abstracted buffer that can encapsulate.

# The `nbuf`

- Use what we like about each
- Linux: Packet data is contiguous in memory
- Linux: Payload data copied into place, headers built out
- BSD: Small header size
- BSD: Few extraneous fields

# The `nbuf`

- We want to convert native to `nbuf`, work in the `nbuf`, and convert from `nbuf` back to the native buffer
- Common-case performance **must** be good
- Copying the data is out

# The `nbuf`

- We can avoid copies if two "fast path" conditions hold:
- Enough space to meet any expansion needs before and after the packet
- Packet data is contiguous in memory

# The `nbuf`

- Expansion space can be arranged for by slightly changing the places data is copied into native buffers.
- We don't do a good job of this right now.
- That turns out not to hurt much.

# The `nbuf`

- Linux: packet data always contiguous in memory
- BSD: usually if `len` $<$ 100 or 208 $<$ `len` $\leq$ 2048.
- Most real IP packets fall into those size ranges.

# The `nbuf`

- We define conversion functions for each system.
- Linux: `skbton()`, `ntoskb()`
- BSD: `mton()`, `ntom()`
- They really do use the fast path most of the time.

# The `nbuf`

- The point of the exercise: Our IPsec ESP and AH code works exclusively with `nbufs` instead of native buffers.
- Common properties makes the code simpler.
- Maybe faster, too.

# Conclusions

- 20% to 40% of the NRL code is system-specific
- 80% to 60% of the NRL code is shared
- Less code sharing than in is common in userland
- Still, that's pretty good

# Conclusions

The point of this talk:

- **Sometimes** it can be done.
- **Sometimes** it's the right thing.
- Set reasonable expectations.
- Systems' maintainers should make it easier.

# Acknowledgments


- The NRL IPv6+IPsec team, past and present
- Ronald Lee, Chris Telfer, Chris Winters, and I did the recent porting work discussed.
- The OS maintainers, esp. those who helped us get code running on their systems.