Networking from the Bottom Up: Device Drivers

George Neville-Neil

gnn@neville-neil.com

May 6, 2009
A Guide to Today’s Class

- Quick Ethernet Overview
- Basic Data Structures
- Break
- Device Startup and Initialization
- Break
- Packet Reception
- Packet Transmission
- Break
- Device Control
- Special Features
Introduction

- Networking begins and ends and the driver layer
- A day in the life of a packet
- Look into many code files in the kernel
- We will use FreeBSD 7.2 (STABLE) as our reference
Device Driver Section Intro

- Lowest level of code in the kernel
- Deal directly with the hardware
- Use a well defined API when interfacing to the kernel
- Are rarely written from scratch
- We will only describe Ethernet drivers in this class
Network Layering

- Application
- Presentation
- Session
- Transport
- Network
- Data Link
- Physical
Network Layering

- Application (All)
- Presentation (Protocols)
- Session (Should)
- Transport (Transport)
- Network (Network)
- Data Link (Data)
- Physical (Properly)
The Four Paths

- Packets traverse four possible paths in the network code
- Inbound (for this host)
- Outbound (from this host)
- Forwarding (between two interfaces on this host)
- Error
Four Paths Through The Stack

- Network Protocol
  - IPv4, v6, etc.

- interface0
  - inbound
  - outbound

- interface1
  - inbound
  - outbound

- forward
Ethernet Overview

- Data Link Layer Protocol
- The most common form of wired networking
- Available in many speeds, now up to 10Gbps
- A simple header followed by data
# Ethernet Packet and Encapsulation

<table>
<thead>
<tr>
<th>Dest</th>
<th>Source</th>
<th>Type</th>
<th>IP Header</th>
<th>TCP Header</th>
<th>Data ...</th>
</tr>
</thead>
</table>
Memory for Packets

- Packets need to be stored for reception and transmission
- The basic packet memory structures are the mbuf and cluster
- mbuf structures have several types and purposes
- Clusters hold only data
- History dictates that mbufs are named m
- In the kernel we will see many pointers to mbufs
Types of mbufs

- Wholly contained
- Packet Header
- Using a cluster
Welcome to SMP

- FreeBSD is a multi-threaded, re-entrant kernel
- Only way to scale on multicore and multi-processor systems
- Kernel is full of cooperating tasks
- Inter process synchronization is required
Kernel Synchronization Primitives

- Spin Locks
- Mutexes
- Reader/Writer Locks
- Shared/Exclusive Locks
- Drivers use mostly spin locks or mutexes
  - See locking(9) for more information
Ethernet Drivers, an Overview

- Implemented in the kernel
  - May be kernel loadable modules (KLD)
- Responsible for getting packets into and out of the system
- Follow a well known set of Kernel APIs
- May drop packets
Introducing, the Intel Gigabit Ethernet Driver

- Supports modern Intel ethernet hardware
- Parts available on motherboards and PCI cards
- A typical example of a modern Ethernet chip
- Driver is well written and maintained by an Intel developer
- A good example to start with
- Data book available at intel.com
- Referred to as igb for short
  - The em driver is the previous incarnation
IGB Features

- Various types of media support
- MSI-X Interrupts
- Jumbo Frames
- Adaptive Interrupt Modulation
- IEEE-1588 (some chips only)
Code Overview

- All FreeBSD device drivers are kept in `/usr/src/sys/dev`
- The IGB driver resides in `/usr/src/sys/dev/e1000/if_igb.[ch]`
- Other supporting files also exist but will not be necessary for this class
- The main data structures are in the header file and the main body of the driver is in `if_igb.c`
- Generic code to support all network drivers is in the `/usr/src/sys/net*` directories
Network Driver Data Structures

- There are two main data-structures in every network driver
  - `ifnet` and `adapter`
- The `ifnet` structure is used to hook the device into the network protocols
- The `adapter` structure is private to the device.
  - The `adapter` structure is often called the `softc`
Objects in C and the BSD Kernels

- Since the early days of the BSDs many kernel data structures have contained both data and function pointers.
- A clever and cheap way to get the benefits of object orientation without paying for unwanted features.
- Function pointers in structures are used throughout the kernel, not just in the network devices.
- No need to be alarmed.
ifnet Overview

- The main interface between the driver and the kernel
- Contains data and functions that are generic to all network devices
- Each device instance must have at least one ifnet
Adapter

- Contains device specific data
  - Hardware registers
  - Device control functions
  - Pointers to packet rings
  - Interrupt vectors
  - Statistics

- Always points back to the ifnet structure
IGB adapter structure
Break

- Please take a 10 minute break
Device Initialization

Relevant APIs

- `igb_attach()`
- `igb_ioctl()`
- `igb_msix_rx()`
- `igb_msix_tx()`
- `igb_msix_link()`
attach()

- Each device driver *must* have a way to connect to the kernel
- The `igb_attach` routine is used to activate a device
- Setup sysctl variables
- Allocate memory
- Set up device registers
- Hook function pointers into place
- Start the device running
Setup Control Variables

- Kernel code can expose controls via sysctl
- Tunables are like sysctls but can only be set at boot
- Used mostly to communicate integers into and out of the kernel
- Also support more complex data structures
Tunables
sysctls
Rings of Packets

- CPU and device share a ring of packet descriptors
- Each descriptor points to a packet buffer
- Used for transmission and reception
- Allows decoupling of the CPU and the device
Packet Ring Structures
Tx Ring Allocation
Allocate Receive Ring
Set Device Registers
Hook in function pointers
Set device capabilities and Media Type
Add Media Types
Start the device
Break

- Please enjoy a 15 minute break
rx()

- Interrupt processing
- Work deferral
- Handling basic errors
- Passing packets into the kernel
Message Signalled Interrupts (MSI/X)

- Old style interrupts required raising a line on a chip
- Old style interrupt routine had to be all things to all people
- MSI allows for different functions to be assigned to different channels
- The IGB driver has one channel per receive or transmit queue and a single interrupt for link state changes
Receive Interrupt
Recieving a Frame
Recieving a Frame (End of Packet)
Packet Reception

Passing in the Packet
Packet Transmission

tx()

- Packets from above
- Work deferral
- Error handling
Protocols Pass Packets Down

- `ip_output()`
- `ether_output()`
- `ether_output_frame()`
- `IFQ_HANDOFF()/IFQ_HANDOFF_ADJ()`
Handing a Packet Off
A word about queues

- Queues of packets are used throughout the networking stack
- Prevent overuse of resources
- Allow for work deferral
- A good way to connect lightly related modules
- Allow the administrator to tune the system
The IGB start routine
Draining the Queue

Packet Transmission
Watchdogs and Drivers

- Hardware is not as perfect as software
- One failure mode is freezing up
- Watchdog routines can be quite harsh
- Continuously resetting a device is *not* the best way to fix it
- Reading `igb_watchdog` is left to the reader
Cleaning up first
Checksum Offloading

- Many protocols required a packet checksum calculation
- Math is hard, and also expensive
- Many 1Gig chips can calculate the checksum in hardware
- For 10Gig this is *required* to operate at full speed
- A layering violation in the stack
Checksum Offload Code
Setup the Transmit Descriptors
Really transmit the packet
Packet Transmission

Break

▶ Please enjoy a 10 minute break
Controlling the Device

- Devices need to be controlled
- Setting network layer addresses
- Bringing the interface up and down
- Retrieving the device state
- The `ioctl` routine is the conduit for control messages and data
Data in/data out
The Big Switch
Setting the MTU
Special Features

- Multicast
- Interrupt Moderation
- Checksumming
Multicast

- One to many transmission
- Mostly handled by hardware
- Table size is important for performance
Interrupt Moderation

- System can easily be overwhelmed by interrupts
- Different types of traffic have different needs
  - Low Latency
  - Average Latency
  - Bulk Transmission
Checksumming

- Difficult to get line rate TCP without hardware help
- Leads to a layering violation
- TCP *must* be aware of hardware checksumming abilities
Section Summary

- All networking device drivers have similar structure
- The hardware details \textit{should} be hidden
- Drivers are rarely written from scratch
  - Copy when write
Questions?