# Implementation of TARGET_MODE applications

BSDCan 2009
Sean Bruno
sbruno@freebsd.org

How we used TARGET_MODE in the kernel to create an interesting product

Kernel Hacking Track

# What Is Target Mode?

- Basic interpretation
  - Turns a BSD box into an external HD via kernel compiler options.
    - Initiator
      - An HBA in hardware or software that accesses resources across a bus.
    - Target
      - A resource on a bus that is accessed by an initiator.
    - Bus
      - SCSI, FireWire, TCP/IP(for iSCSI), ATA over Ethernet,
      - Fibre Channel, Fibre Channel over Ethernet
      - The media used in Initiator-Target communication

# Example Targets In Use

- MiraLink Product Lines (Shameless Plug)
  - Acts like a hard drive
  - Intercepts blocks and buffers them
  - Copies blocks to duplicate unit
    - Completely Agentless
  - Use your own hard disk
  - Fiber Channel volume “import”
  - Easy to use and “abuse”
    - More on this later

# Example Targets In Use

- Mac Book FireWire Target Mode
  - On Power up, hold the letter 'T' until FireWire logo appears
  - Now it's a read-only FireWire drive enclosure
  - You can access the DVD drive across FireWire
- Linux iSCSI Target
  - Software server
  - Supports multiple targets
  - Support multiple accesses

# How Do I Do That?

- Requires some kernel tweaks
  - Targ(4)
  - driver specific kernel options
  - AIO(4) support for scsi_target example code
- Requires something to be used as a target
  - You still need a disk or file to use as your target device to be presented on the bus
- Appropriate BUS interface board.
- Protocol specifications for your BUS
  - e.g. SBP-2, SAM-2

# Is That All?

- A good mentor, I've had several awesome ones:
  - Justin Gibbs, Scott Long, Hidetoshi Simokawa
  - freebsd-scsi@, freebsd-firewire@ mailing lists
  - freebsd-hackers@ if you are feeling dangerous.
- Don't Panic.
  - Your machine will do it for you.

# How does that work again?

- Kernel Configuration
  - TARG(4): The Interface to CAM
    - You'll need to enable.
    - It's where the “magic” happens.
    - Provides a fake device to attach or open
      - e.g. /dev/targ0
    - Gives your application access to “raw” data
    - Application must implement target protocol
      - e.g. SAM-2, SBP-2, SAS
    - Link Layer protocols are handled by card/driver
      - FireWire board and FireWire driver handle the data
      - You must do something compliant with SBP-2
    - Read the man page, it's got good stuff!

# How does that work again? (cont)

- Kernel Configuration
  - TARGBH(4): Black Hole
    - Allows non-existent targets to be NACKED
    - Makes the target play nice on the BUS
    - Provides a fake device to attach or open
      - e.g. /dev/targbh0
  - AIO(4)
    - Required kernel option by scsi_target
    - May be ok to leave out, but I sure haven't tested without it.  You mileage may vary.

# How does that work again? (cont)

- Kernel Configuration
  - Choose your target interface
    - aic7xxx U160 SCSI
    - qla2342 2G Fibre Channel
    - qla1040 or other Qlogic SCSI boards
    - Random FireWire Adapter
  - Each driver has it's own target mode flags
    - Or is a separate driver on it's own, e.g. sbp_targ

# Drivers, Drivers, Drivers

- ahc(4)
  - Options AHC_TMODE_ENABLE XXX
  - 0xNN is a bitmask of the the units you want to activate
    - 0x25 enables unit 0, 2 and 5 for target mode.
    - 0x8a enables units 1, 3 and 7 for target mode.
    - Go ahead, convert it to binary, I'll wait.
  - Once enabled, different firmware is loaded
  - SCSI card behaves differently

# Drivers, Drivers, Drivers, cont.

- isp(4)
  - Options ISP_TARGET_MODE
  - No magic bitmask here
  - Once enabled, different firmware is loaded
  - 2G and old school qla1040 SCSI cards supported
    - Probably intermediate chipsets are supported as well
    - No.  I am not holding some magic 4/8G code that works
    - Qlogic changed interface for 4/8G chipsets

# Drivers, Drivers, Drivers, cont.

- Isp(4) ... cont
  - Qlogic changed interface for 4/8G chip sets
  - Overall, this is a good thing
  - More control for targets and initiators in the host operating system

# Drivers, Drivers, Drivers, cont.

- FireWire(4)
  - FireWire has a lot of parts
    - sbp(4) is the initiator, this should be disabled
    - sbp_targ(4) is the target, this should be enabled
    - All other normal firewire drivers should be enabled
      - firewire, fwohci etc.
    - Sbp(4) and sbp_targ(4) might work together.  Untested
  - 200/400/800 should work fine in -CURRENT

# Drivers, Drivers, Drivers, cont.

- Stuff I'm Ignoring
  - Target code exists in MPT(4)
  - NetBSD software iSCSI Target
  - Proprietary targets
  - Embedded USB Device Controllers
- We can talk later about these items.

# What Could Possibly Go Wrong?

- Serial Console
  - You'll need it.  I promise.
- Get friendly with the debugger
  - You'll be meeting it very frequently
- Be patient
  - You'll be rebooting a lot
- My FireWire Kernel Configuration
  - Http:///

# share/examples/scsi_target

- Userland code
  - Great example of how to get started
  - Instructions right in the man page.
  - Create a dummy file with dd if=/dev/zero
  - Compile scsi_target
  - Run it as root
    - ./scsi_target 0:0:0 /var/tmp/myfile
  - Now, connect to a new machine.
    - If all went well here, I can stop sweating.

# SBP-2 and SAM-2

- SBP-2 is a SAM-2 like protocol
  - Not a fully featured as SAM-2
  - Really, only for storage devices
  - Close enough and the protocol is freely available
  - Unlike the rest of the IEEE FireWire protocols
- SAM-2
  - Big protocol specification
  - More Features and interfaces supported
  - Draft standards are available on the T10 committee web site (ANSI)

# It's just an external hard drive

- Seriously, that's all we've done here.
- More code coming down the pipe
  - Enhancements to scsi_target
  - Overhaul of sbp_targ is imminent
  - Enhancements to sys/cam/scsi_target
- Once you have access to the data
  - Interesting ideas come to you
  - e.g. block level snap shots, backups completed by the drive instead of the host

# Future Development

- Multiple LUN Support
  - Single interface to multiple targets
- Multiple simultaneous target support
  - Fibre Channel and FireWire target at the same time
- Simultaneous SBP and SBP_TARG use
- iSCSI Software Target
- USB Mass Storage Hardware Target
  - Needs USB Device Controller Support

# Special Thanks

- Justin Gibbs – many long hours of questions
- Scott Long – my src commit mentor
- Hidetoshi Simokawa – FireWire guidance
- Dan Langille and the BSDCan folks!
- Matt Jacobs, ISP
- FreeBSD Foundation

# Questions?

- Keep 'em simple, I'm not as smart as I am pretty.