Case Study - Embedded Linux in a digital television STB

Melanie Rhianna Lewis

About me (1)

- Software Consultant for RED Embedded Consulting in the UK
- 15 years of Linux experience
- 10 years of Digital TV experience
- Major clients in the UK, US and elsewhere

About me (2)

- Also...
  - PHP PECL maintainer
  - Active in the open source community (PHP Women, Devchix, Ubuntu Women...)
  - Maker/Crafter
  - Bellydancer, skater, biker...

Introduction (1)

- Case study of Embedded Linux in a digital television set top box :-)
- Chosen a Broadcom / MIPS satellite DVR
- Some details may be omitted due to NDA issues
- Any opinions are my own and not my employers

Introduction (2)

- Commercial realities of set top boxes
- Digital television transport protocols
- What's in a set top box?
- A typical software stack
- Linux components of the stack
- Driver architecture and performance

Commercial realities of set top boxes

- The customer is the TV broadcaster/distributor, not the end user
- The customer often has strict control over software versions and components
- The customer’s business is selling subscriptions, content and advertising
- DRM is key :-(

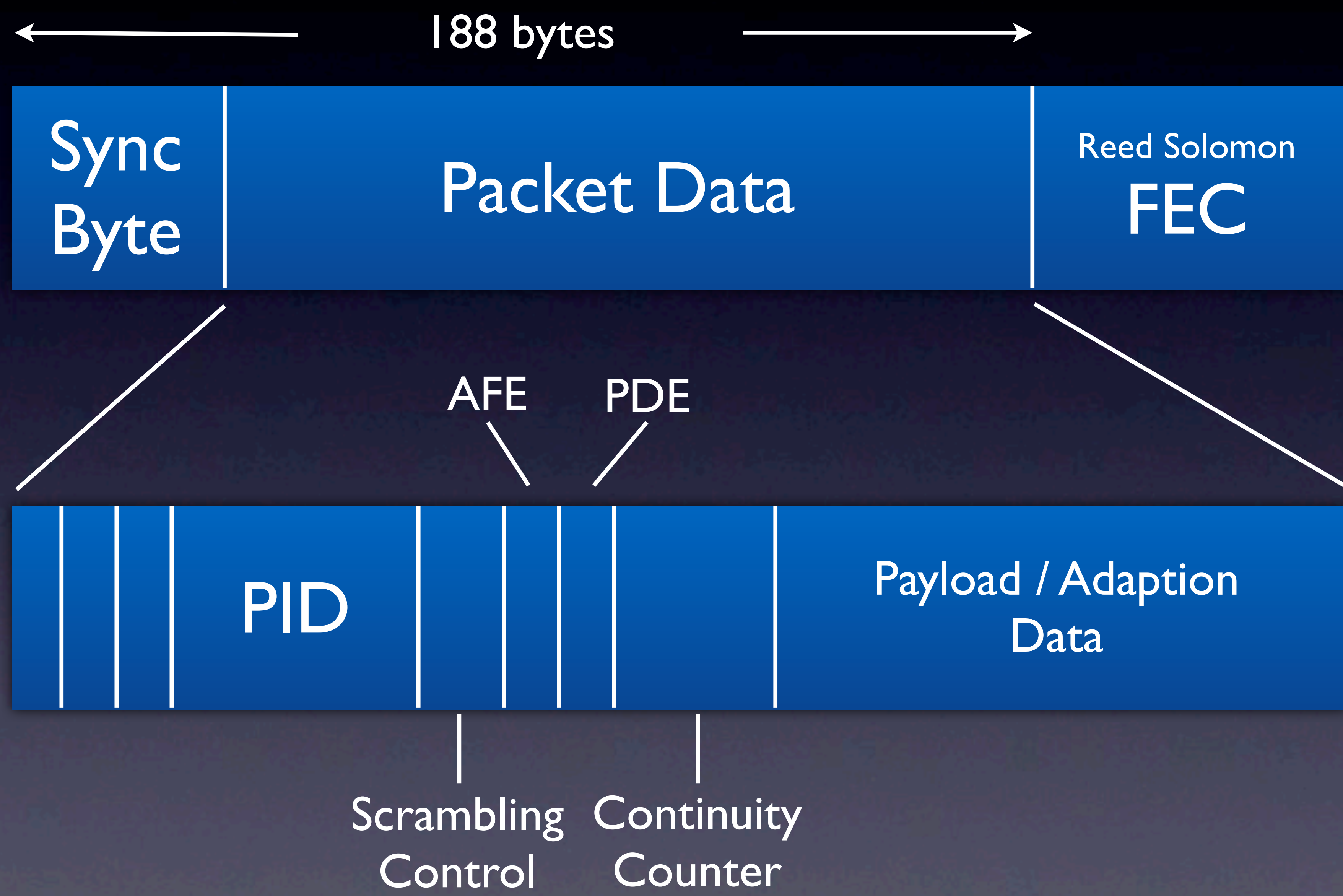
Digital Television Transport Protocols

- Primarily based on extended MPEG-2 protocol
  - DSS (Direct Satellite System) - Proprietary
  - DVB (Digital Video Broadcast)
  - ATSC - (Advanced Television Standards Committee)

DVB

- The DVB standard has variants for cable, satellite and terrestrial.
- Packetised stream
- Extensions for Bouquets, Services, Events, Time and Date, etc.
- Extensions for CA management (EMMs, ECMs, etc)

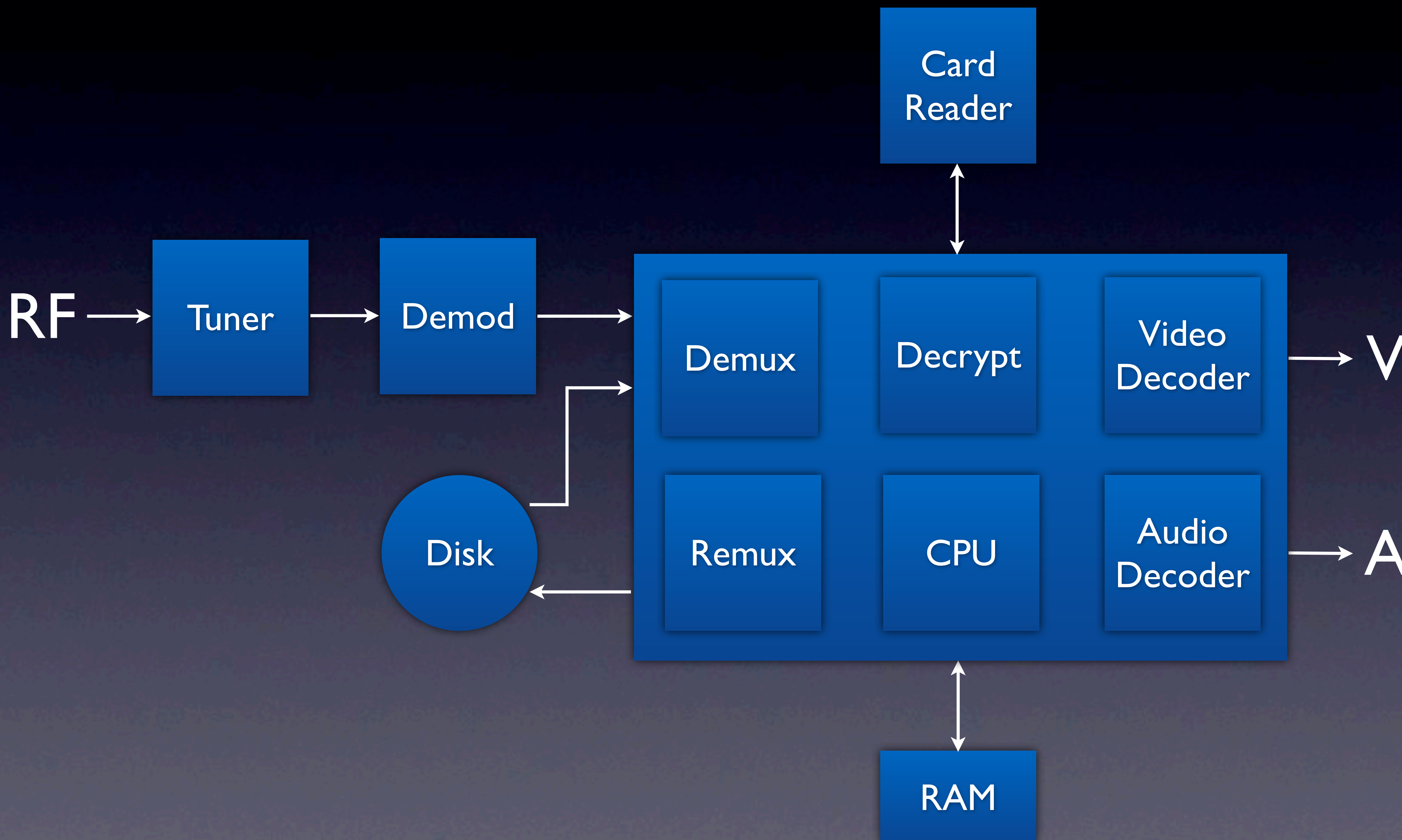
DVB


188 bytes
Sync Byte
Packet Data
Reed Solomon FEC
AFE
PDE
PID
Payload / Adaption Data
Scrambling Control
Continuity Counter

DVB

- Program Allocation Table (“PAT”) - PID 0x0000
- Conditional Access Table (“CAT”)- PID 0x0001
- NULL packets - PID 0x1FFF
- Program Map Table (“PMT”)
- Network Information Table (“NIT”)
- Video/Audio
- Program Clock Reference (“PCR”)

What's in a set top box



What’s in a set top box

- Almost a single chip solution using a SOC
  - Broadcom / MIPS
  - Trident / ARM (was NXP)
  - ST / SH4
- Few other components such as tuners / front panel etc.

The SOC (1)

- Core processor (MIPS, ARM, SH4)
- Video/audio decoder
- Media processor/DSP
- Graphics core (2D/3D)
- SI filtering
- Encryption/decryption core

The SOC (2)

- Media data paths are end to end i.e. from the demodulator to the video
- The media drivers manipulate the media cores not the media
- If media leaves the SOC (for example on a DVR) then typically it is re-encrypted
- Most user space data processing is of the meta data such as guide data etc.

Set top box statistics

- Typically...
  - 400MHz core processor
  - 256MB RAM
  - 128MB FLASH (NOR and/or NAND)
  - 64KB EEPROM
  - 0.5TB Hard disk

The Kernel and FS

- STB must work without HD so kernel and filesystem on flash
- If just NOR
  - Kernel stored in mtd partition
  - Root file system in squashfs in mtd partition
- If NOR / NAND or just NAND
  - Kernel and root filesystem as initrd image
- Flash/HD used for data storage

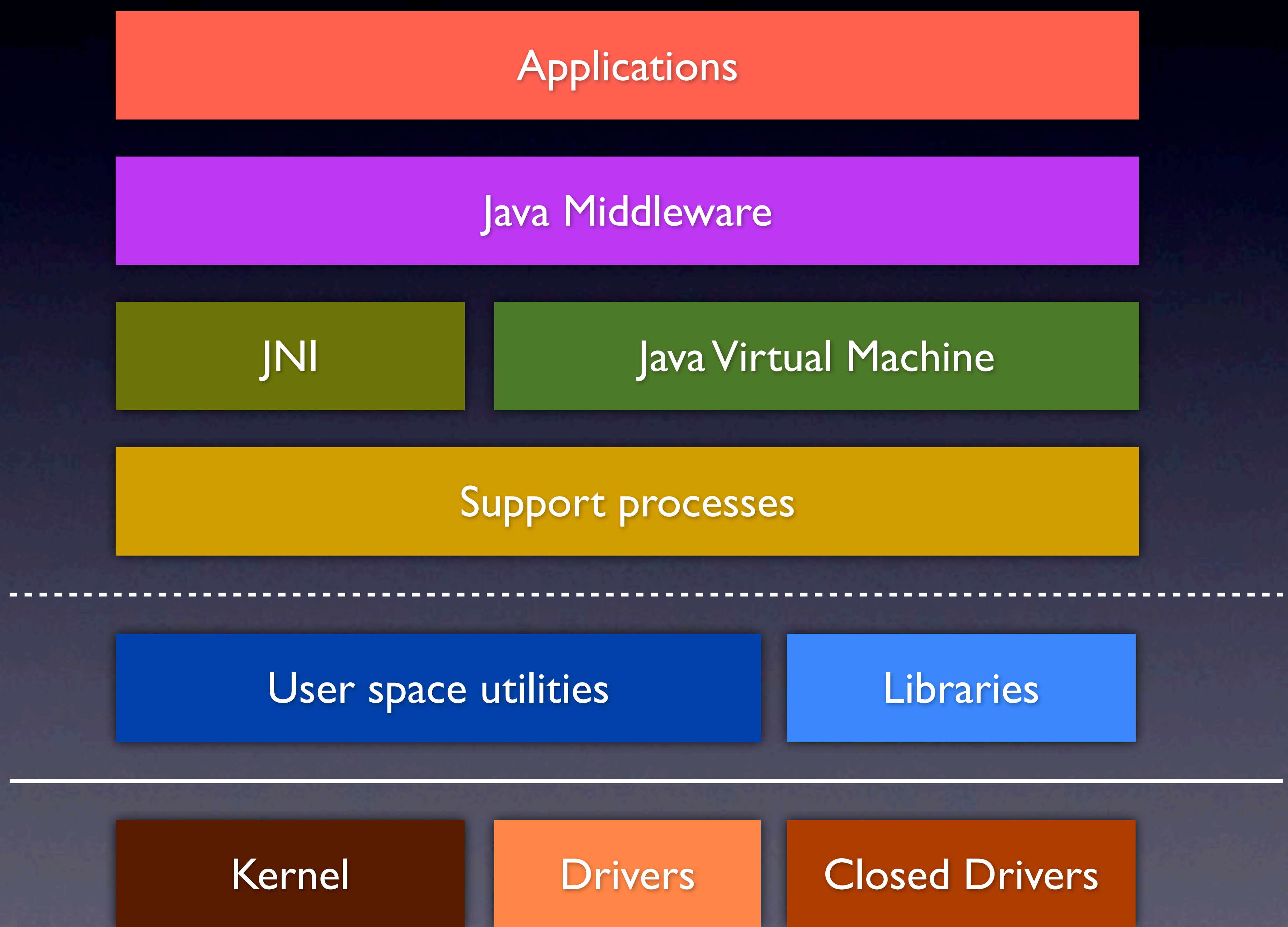
Boot Loader

- In production uses proprietary loader than can download new platform image over air
- Typically two images (one for backup)
- In development use developer bootloader
  - MIPS - CFE (Developed by Broadcom but open source)
  - ARM / SH4 - U-Boot (Open source)

Development

- Use development boot loader
- tftp for the kernel
- NFS for the file system
- Serial console / ssh
- Add additional utilities such as...
  - gdbserver
  - ldd
  - strace

Software Stack



SOC vendors

- Don't just sell the silicon
- Provide the extended kernel and toolchain
- Provide the driver library for media processor (E.g. Broadcom's Nexus and STMicroelectronic's STAPI)
- For STBs they will provide drivers written to the customer's required driver API (E.g. NDS' CDI API)

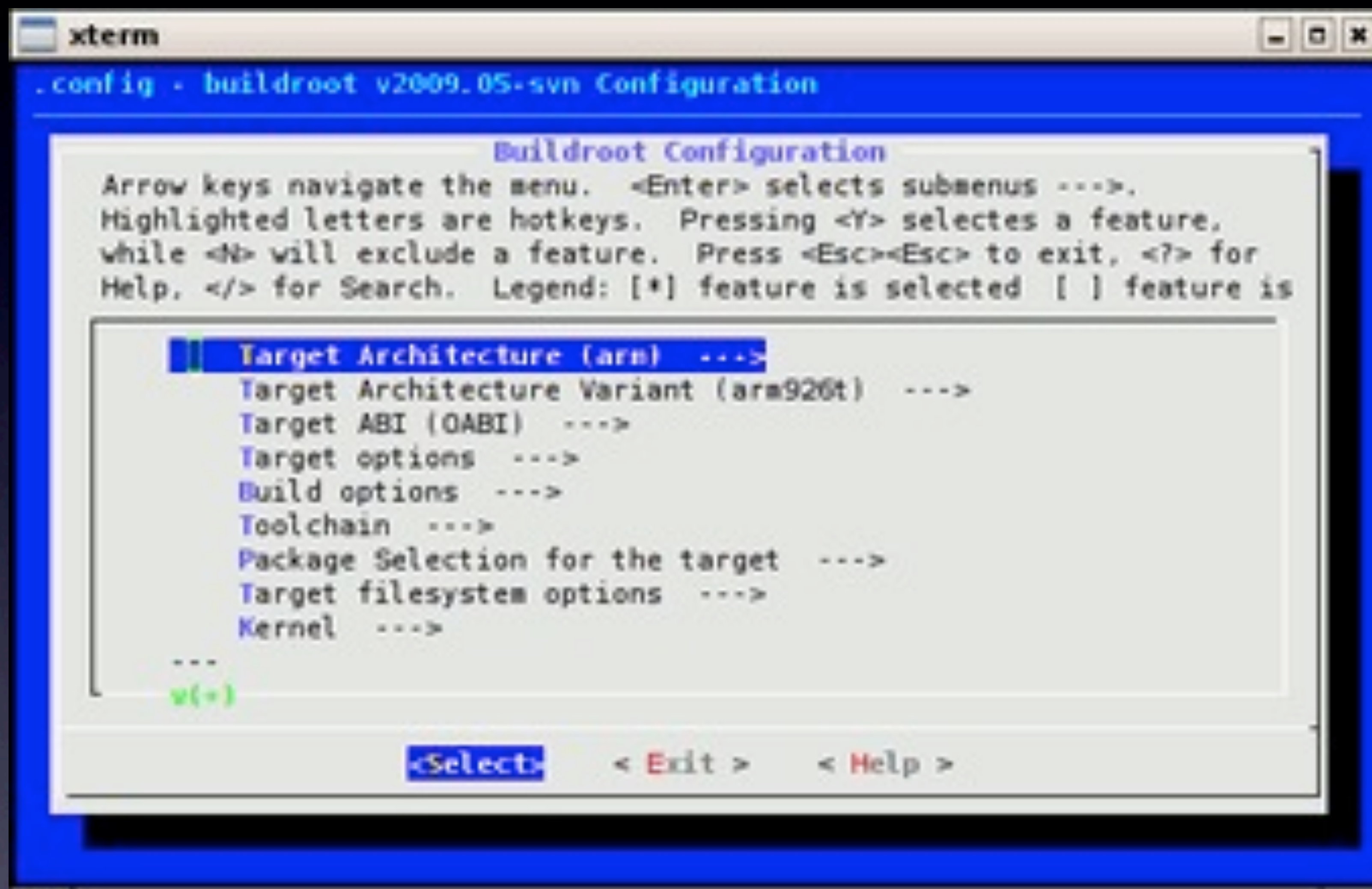
Kernel

- Based on vanilla linux.org kernel but extended...
  - Architecture support
  - TTY, I2C, USB host, Ethernet drivers etc
  - Back ports of security fixes
  - RT extensions
  - Possibly core changes
  - DOES NOT include media drivers

Buildroot (1)

- Builds a kernel and root file system from scratch
    - Uses a ‘menuconfig’ style config mechanism.
    - Cross compile tools
    - Kernel
    - uClibc and other libraries
    - Command line utilities and debugging tools
    - Root file system as tgz, initrd, squashfs etc.

Buildroot (2)



http://buildroot.uclibc.org/

Buildroot (3)

- Possible buildroot changes to support chip vendor's kernel and tools
  - Add platform support for STB
  - Add kernel patches
  - Add toolchain patches
  - Add uClibc patches
  - Add build support for any additional utilities

Open Source Components

- Kernel
- uClibc
- Busybox
- BASH
- pppd
- ldconfig
- SQLite
- strace
- gdb
- fsck
- hdparm
- portmap
- rpc.mountd
- rpc.nfsd

Not exhaustive

Media Processor Drivers

- The media processor drivers are typically propietary and support
  - Demodulator
  - De/remultiplexor
  - SI filtering
  - De/encryption
  - Video/Audio decoding
  - Graphics
- Built independently of the open source code.

The Proprietary Userspace Code

- Conditional access library/process (Supplied as a binary)
- Java virtual machine (Supplied as a binary)
- Support processes
- Java layer

Support Processes

- Utility applications written in C/C++
- Performance critical tasks
  - Meta data processing
  - Media processing
  - Reading/writing media data to HD
- ‘Glue’ between Java and Kernel/Libraries

Java Layer

- Guide applications
- UI
- Middleware engine. Manages...
  - Setup and install
  - Program guide data
  - Recordings
  - Background download
  - Software update

Driver Architecture (1)

- Kernel space drivers
  - Proprietary kernel space drivers.
- User space drivers with a kernel space event driver
  - Event driver passes interrupts up to the user space code
  - Registers mmap-ed in to user space

Driver Architecture (2)

- Never seen a combination.
  - Why? Put performance critical in kernel and the rest in user space
- Pet peeve - IOCTLs are NOT an API
  - Always, always have a user space library on top of the IOCTLs.
- In the end the 'API' is usually defined by the middleware vendor. This determines the driver architecture.

Driver Architecture (3)

- Most drivers deal with hardware configuration
- Devices represent a logical model
- Use DMA for transferring media and meta data.

Threading

- RT threads should not be needed in user space.
  - RT critical code should be in the kernel
  - Have seen poorly designed code that needed priority ‘tweaking’
- Are RT threads needed at all?
  - Most processing in HW. Software just configures
  - SOC vendor defines requirement for RT threads

Any questions?

Where to find me

Web - http://www.cyberspice.org.uk/

Blog - http://www.cyberspice.org.uk/blog/

Twitter - @Cyberspice

LinkedIn - MelanieRhiannaLewis