Linux in a Light Bulb

How far are we on tinification?

Pieter Smith
Philips Lighting
The humble light bulb

Most under-appreciated appliance in your home
A light bulb is...

- Ubiquitous
- Used daily
- Largely unnoticed
  - Unless it is *broken*
Why connect a light bulb?
Affects your biology

6:00 AM Wakeup.

10:00 AM at work.

3:00 PM at work.

11:00 PM Bedtime

Oh, now I'm awake. Thanks a lot body.
Affects your biology

- Circadian rhythm
- Treatment of *sleep* disorders
Affects your mood

- Ambiance creation
- Entertainment
Affects perception of safety / security

- Soft security
Gentle reminders

- Alarm clock
- Door bell
- Weather status
Tunability
Connecting things

• Traditional approaches:
  – Add a gateway
    • Simple nodes (E.g. Zigbee)
  – Get a bigger SoC
    • Direct IPv4/6 connection to internet

• Not what SoC vendors are advocating
  – With some exceptions
SoC vendors

- Pushing cost / feature
  - Driven by functionality
    - E.g: WiFi @ +$1 (BOM)
  - Networking stack in on-die ROM
  - RAM / NOR secondary
SoC vendors

• NOR flash
  - Some vendors moving NOR off-die
  - Multi-channel SPI NOR
  - XIP via smart peripheral + instruction cache

• RAM
  - Slow to increase
Internet of “broken” things

- Proprietary stacks
  - Not open to public scrutiny

- Security
  - RAM patching of ROM stacks
    - RAM and NOR flash needs to be reserved
    - Lack of liability + cost pressure
  - Security is a **process** not a state
    - SoC vendors traditionally slow to respond
Why Linux is better?

- Best networking stack
- Best driver support
- Huge test-surface
- Developer mind-share
- Open-source (Auditability)
- Security process
Challenges: Price point

- Samsung Galaxy S6 @ €570
  - SoC + RAM + FLASH @ €73
  - Easily runs Linux
- Home router @ €100
  - SoC + RAM + FLASH @ €10
- Connected LED light bulb
  - Color @ €60
  - White @ €30
Challenges: Thermal design

- Internals run at 100 °C when $T_A = 40 ^\circ C$
  - 10 W rating (LEDs + Power electronics)
  - Small housing

- The chosen SoC must:
  - Operate @ 125 °C
  - Have low power consumption
    - Don't generate *more* heat
What do we need from Linux?

• Tiny size:
  - Small SoC
A brief history on kernel size

Linux on a *floppy*-disc:

- 2001: v2.2.19 @ 977KB compressed
- 2004: v2.4.27 @ 797KB compressed
- 2004: v2.6.8 @ 1073KB compressed
A brief history on kernel size

- 2001: v2.2.19 @ 977KB compressed
- 2004: v2.4.27 @ 797KB compressed
- 2004: v2.6.8 @ 1073KB compressed

- 2015: v4.2 @ 5.8 MB compressed (defconfig)
  - Not an honest comparison
Possible causes for kernel bloat

- (Intentionally) prioritize developer efficiency.
- Unnecessary / badly designed abstractions.
- Code duplication.
- **Unused feature accretion.**
How about the tiny use-case

- defconfig not so useful for tiny systems
- Let's compare history of tiny configs
Tiny mainline kernel

- Create .config template with only:

  ```
  CONFIG_EMBEDDED=y
  CONFIG_EXPERT=y
  CONFIG_CC_OPTIMIZE_FOR_SIZE=y
  CONFIG_KERNEL_XZ=y
  CONFIG_OPTIMIZE_INLINE=y
  CONFIG_SLOB=y
  CONFIG_NOHIGHMEM=y
  ```

- Run:

  ```
  make KCONFIG_ALLCONFIG=${path_to_above} allnoconfig
  make
  ```
vmlinux dissected

.text
- Constants and code
- Can remain in directly addressable FLASH

.data
- Initialized variables
- Has to be copied from FLASH to RAM

.bss
- Uninitialized data
- Only occupies RAM
How much RAM and FLASH?

• For XIP (Execute in-place):
  - .text + .data => FLASH
  - .bss + .data => RAM

• For compressed kernel image:
  - bzImage => FLASH
  - .bss + .data + .text => RAM
XIP versus Compressed Image

- **XIP:**
  - Saves RAM at the expense of FLASH
  - FLASH must be directly addressable by CPU

- **bzImage:**
  - Saves FLASH at the expense of RAM
  - No special addressability needs for FLASH
How far are we on Tinification?

Kernel size history (XIP)

- bss+data (RAM)
- data+text (ROM)
Linux in a Light Bulb

How far are we on Tinification?
A brief history of the kernel weight-watchers

- Linux yo-yo diets
Enter linux-tiny

- 2003: Started by Matt Mackall
  - First patch-set for v2.6.0
- 2005/2006: CELF sponsorship
  - Top 17 patches mainlined
Dither linux-tiny

- 2006: Mostly abandoned
- 2007: Revived by CELF
  - Michael Opdenacker volunteers
  - http://elinux.org/Linux_Tiny
  - http://elinux.org/Kernel_Size_Tuning_Guide
Wither linux-tiny

- 2007: Last patch release @ v2.6.23
- 2008: Focus only on mainlining
  - Most promising (51) patches only
- 2008: Mailing-list archive ends

- Today: 2 / 51 patches mainlined
Bloatwatch

• 2006: Matt Mackall
  – Written at CELF as size regression tool
• Today https://www.selenic.com/bloatwatch/:
  “This project has been discontinued due to lack of cooperation from kernel.org admins.”
Enter Linux kernel tinification

• 2014: Josh Triplett
  – Call for arms at ELCE 2014

• Topics:
  – Making more of Linux *optional* (E.g. perf)
  – Link-time optimization
  – Automatic syscall elimination
  – Mainline OpenWRT tinification patches
  – GCC improvements for size reduction
Linux kernel tinification

- v3.18 merge window
  - Maintainer gripes
  - Merge conflicts
- Let things cool down:
  - Skip v3.18
  - Retry at v3.19
Dither Linux kernel tinification

• So Josh just has to wait 60+ days, right?
Dither Linux kernel tinification

• So Josh just has to wait 60+ days, right?
• Day-job
  – Chrome OS Architect @ Intel
• Other cool projects
  – clonefd
  – **BITS**
  – Both presented at LinuxCon 2015
• Mainlining stalled
Not so glum...

• Some patches mainlined:
  – E.g. `fadvise()` / `madvise()` now optional
• Number of patches posted for review
• Tools to hunt for bloat are getting better
Comparison with PREEMPT_RT

- 2004: First patch-set in by Ingo Molnár
- 2004: Thomas Gleixner picks up top of tree
- Stable picked up by Steve Rostedt
PREEMPT_RT

• Parts with general value mainlined
• RT-specific parts require nurturing into mainline
  – Rewrites
  – Show non-RT value
    • While solving RT problems
• Effort already > 10 years and still going strong
  – Real-Time Linux Collaborative Project
How should we proceed?

- Have patience
- Coordinate efforts
  - Consider partnering up with other tiny use-cases
How should we proceed?

- Have patience
- Coordinate efforts
  - Consider partnering up with other tiny use-cases
- Tips from Linus Torvalds and Thomas Gleixner:
  - Improve existing code
  - Demonstrate mainline value first
  - Slip stuff in in small increments / nicely disguised Trojan horses
  - Sell crazy stuff using non-crazy arguments
Improved tinification approach: An example

Assisted Link-time Optimization

• Remove redundant symbols from ELF's
  – Identify exported symbols not needed by compiled modules
  – Analyze user-space for syscall dependencies

• Let link-time optimization remove the rest

• Kernel work
  – Fix things that break the above:
    • E.g. Implicit / unneeded dependencies (General quality)
Back to the connected light-bulb
My roadmap

Ethernet / (Wifi)
Sep 2015

Ethernet / (Wifi)
ELCE2015

Wifi
20xx
Linux in a Light Bulb

How far are we on Tinification?

Ethernet / (Wifi)
Sep 2015

Ethernet / (Wifi)
ELCE2015

Wifi
20xx

You are here
The set-up

- **Beaglebone Black**
  - Tiny Linux kernel
  - Tiny Userspace
    - Connected application
- **Lamp**
  - Power electronics
  - LED's
  - PWM
The application

- Something ELCE doesn't yet know they absolutely **NEED**

- **The trivia-bulb**
  - Self-hosted web trivia
  - Technicolor effects
Let's hack a trivia-bulb

```bash
git checkout v4.2.3
make tinyconfig
make ARCH=arm CROSS_COMPILE=arm-buildroot-linux-uclibcgnueabihf-
```
## From tinyconfig to something useful

<table>
<thead>
<tr>
<th>config</th>
<th>zImage</th>
<th>RAM</th>
</tr>
</thead>
<tbody>
<tr>
<td>tinyconfig</td>
<td>284 KB</td>
<td>777 KB</td>
</tr>
<tr>
<td>+mm</td>
<td>329 KB</td>
<td>865 KB</td>
</tr>
<tr>
<td>+thumb2 kernel</td>
<td>433 KB</td>
<td>759 KB</td>
</tr>
<tr>
<td>+networking</td>
<td>563 KB</td>
<td>1012 KB</td>
</tr>
<tr>
<td>+initramfs</td>
<td>571 KB</td>
<td>1022 KB</td>
</tr>
<tr>
<td>+SoC (AM33XX)</td>
<td>660 KB</td>
<td>1227 KB</td>
</tr>
<tr>
<td>+ethernet/edma/phy drivers</td>
<td>679 KB</td>
<td>1264 KB</td>
</tr>
<tr>
<td>+tty/serial drivers</td>
<td>783 KB</td>
<td>1476 KB</td>
</tr>
</tbody>
</table>

Real world isn't that simple
Complications (Challenges)

• Kernel does not boot
  - Goes quiet after u-boot handover
  - Early printk does not help
  - omap2plus_defconfig doesn't either

• First get it working
  - Back to buildroot defaults
  - Vendor kernel: git://git.ti.com/ti-linux-kernel/ti-linux-kernel.git
  - Config: board/beaglebone/linux-3.12.config
Getting it working

- Great: Buildroot default works
- Strip out all we don't need
- Subtle dependencies:
  - Serial port needs RTC + watchdog to function
- Non-explicit configuration dependencies
- Large static RAM instantiations in kernel/lockdep.c:
  - lock_classes: 2.1 MB
  - stack_traces: 1.0 MB
How about userspace?

• Single ELF binary to replace init
  – Implements trivia application
  – Handshake with light-bulb and drive PWM's

• Statically linked:
  – Onion web-server library: https://github.com/davidmoreno/onion
  – Uclibc (Not always allowed)
Linux in a Light Bulb

How far are we on Tinification?

Results are hideous

<table>
<thead>
<tr>
<th>config</th>
<th>zImage</th>
<th>RAM</th>
</tr>
</thead>
<tbody>
<tr>
<td>trivia bulb BSP</td>
<td>1.7 MB</td>
<td>8.6 MB</td>
</tr>
</tbody>
</table>

But this is a work in progress

*With a roadmap ending in 20xx*

Most importantly:
This is an example of what you might run into
Next steps

git clone https://git.kernel.org/pub/scm/linux/kernel/git/torvalds/linux.git
cd linux
git remote add linux-next \\https://git.kernel.org/pub/scm/linux/kernel/git/next/linux-next.git
git fetch linux-next

git fetch --tags linux-next

# Get it working / fix stuff

git commit
git format-patch
git send-email
Next steps

- Switch to a Cortex-M3/4/7 development kit
- Rinse and repeat
Please participate

- Connect your smart device to lfevents
- Open your web-browser
- Answer all questions
  - The first *est* wins

10.200.66.18
References

- Sources and work associated with this presentation
  - https://github.com/smipi1/elce2015
- Linux tiny
  - https://lwn.net/Articles/608945/
  - http://lwn.net/Articles/63516/
  - http://elinux.org/Linux_Tiny_Patch_Details
- Linux tinification effort
  - https://tiny.wiki.kernel.org/start
  - https://lwn.net/Articles/608945/
- Size tuning
- Tips
  - https://lwn.net/Articles/370998/
Acknowledgements

- Josh Triplett
- Thomas Gleixner
- George Yianni
- Adriaan van den Brand
- Hue development team @ Philips Lighting