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***

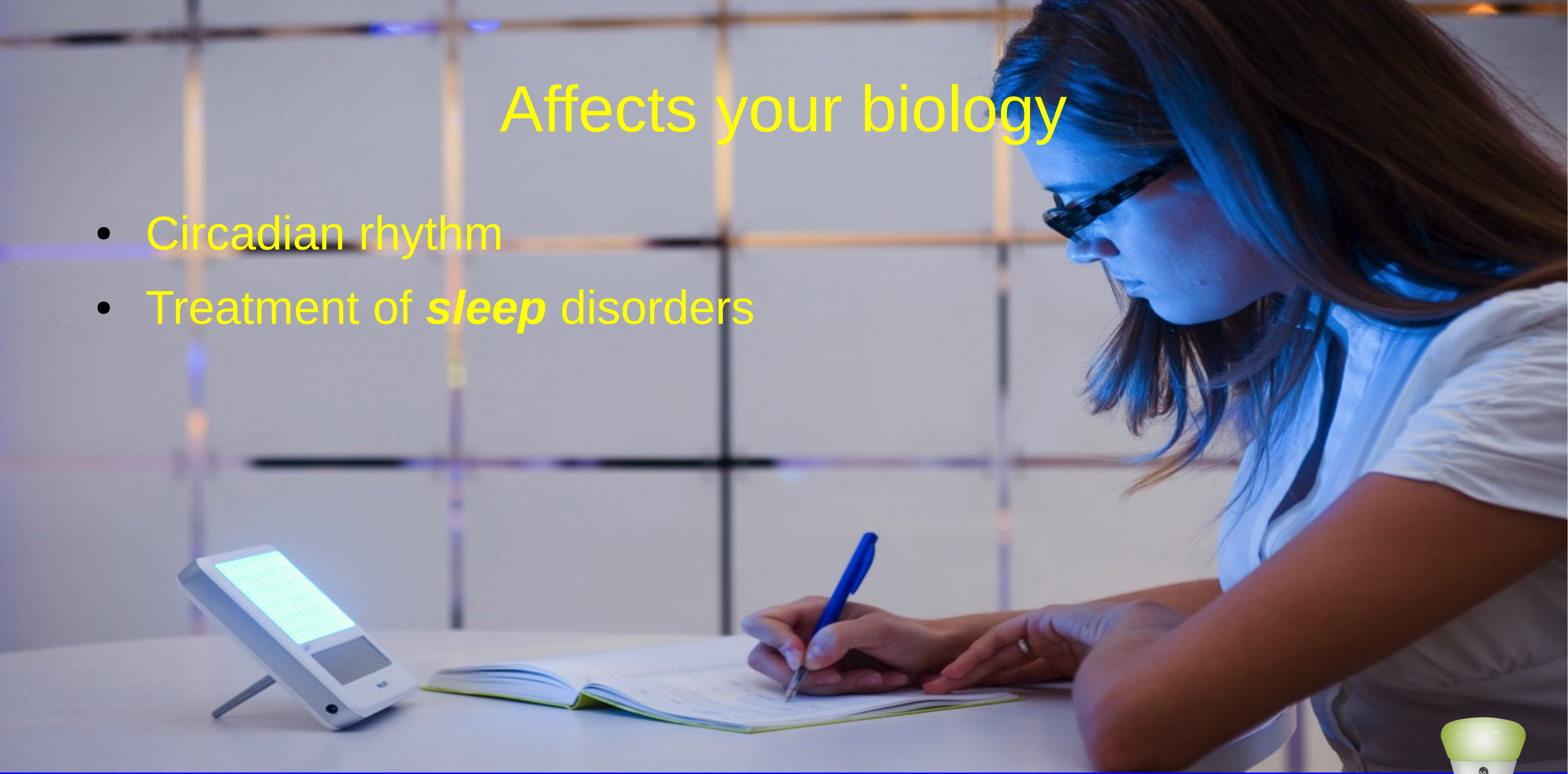
Why connect a light bulb?

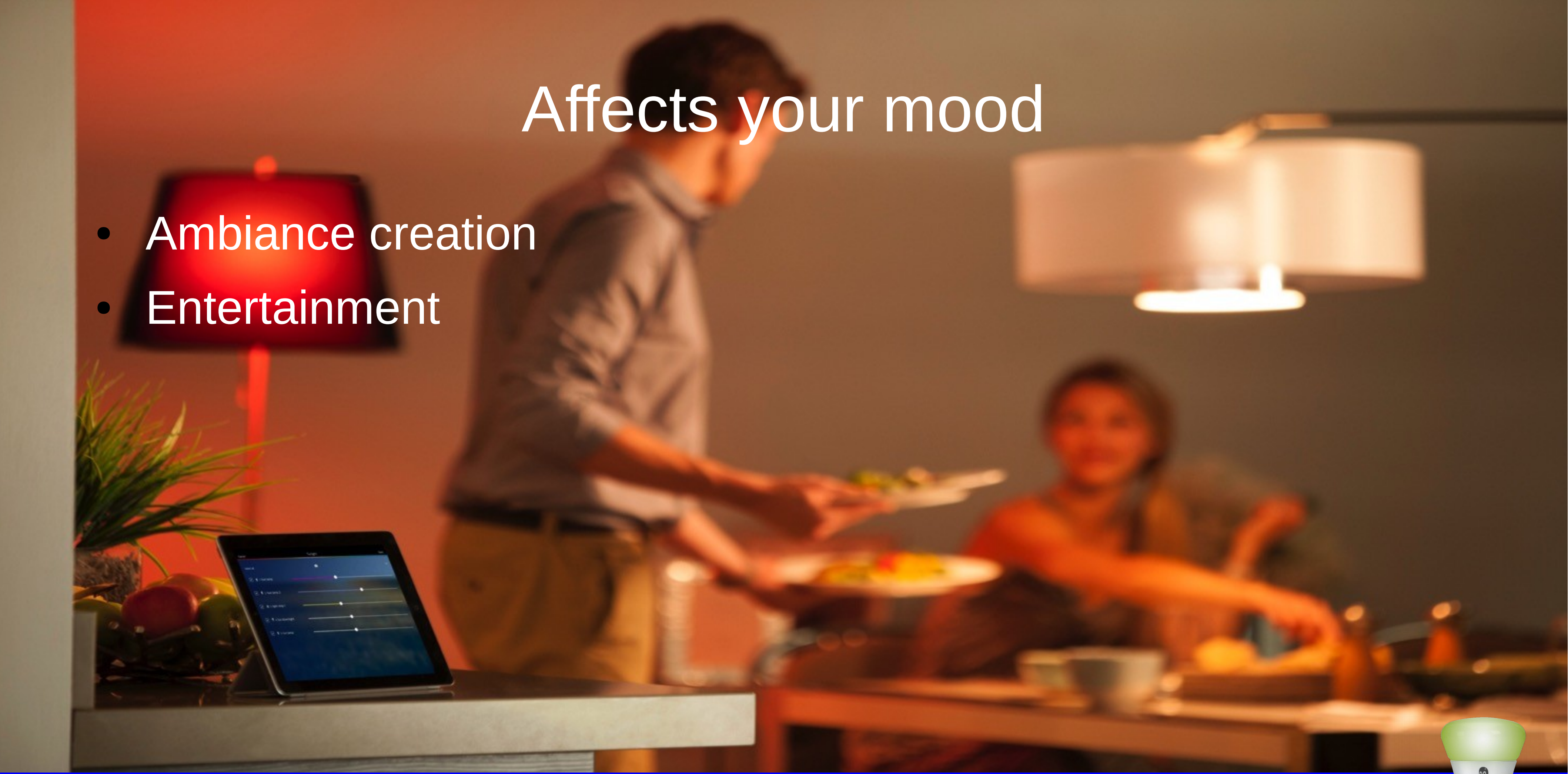
Affects your biology

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

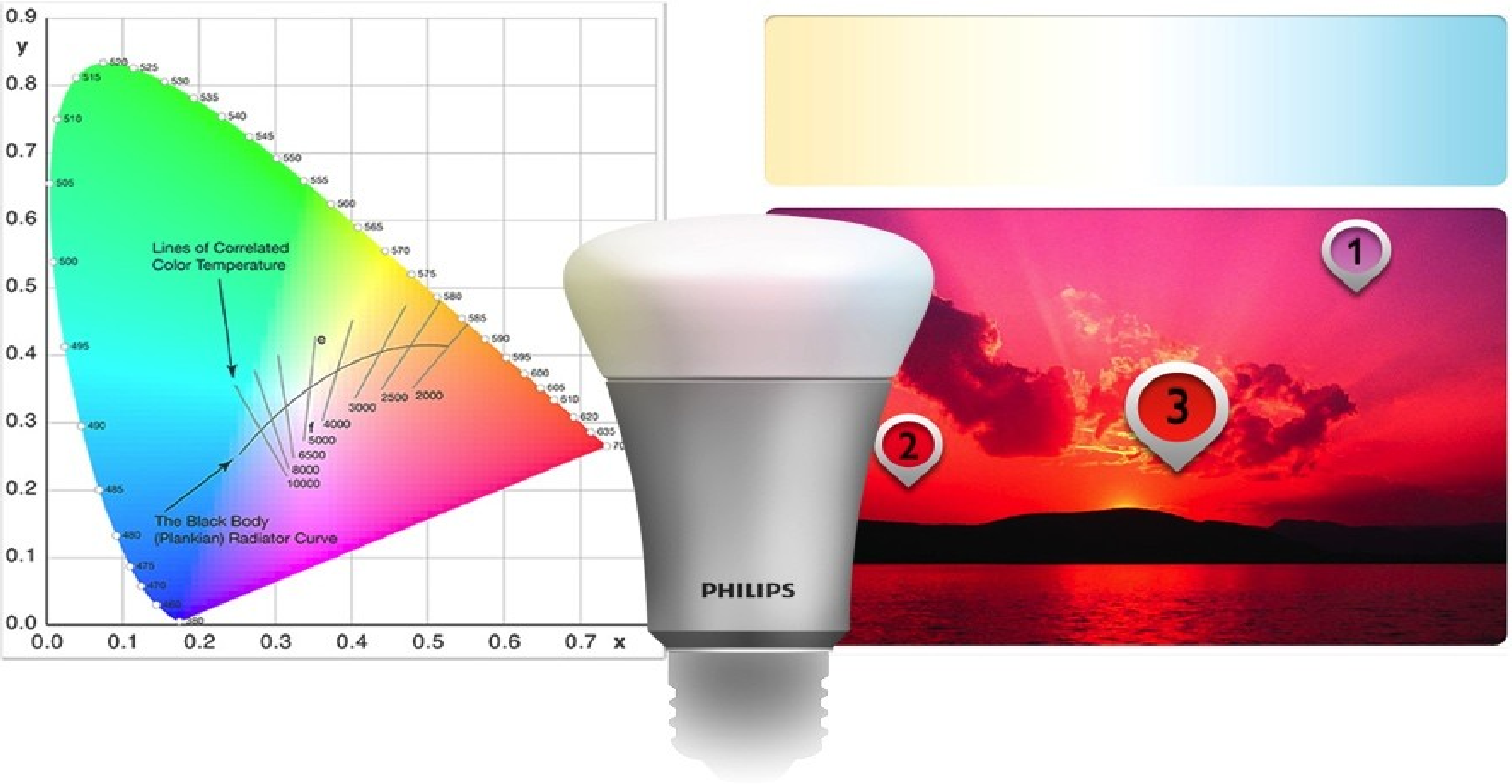
Affects perception of safety / security

- Soft security

Gentle reminders

- Alarm clock
- Door bell
- Weather status

Tunability


0.9
y
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0.0
0.0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
x
380
460
470
475
480
485
490
495
500
505
510
515
520
525
530
535
540
545
550
555
560
565
570
575
580
585
590
595
600
605
610
620
635
700
Lines of Correlated Color Temperature
The Black Body (Plankian) Radiator Curve
e
f
10000
8000
6500
5000
4000
3000
2500
2000
PHILIPS
1
2
3

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 °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 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_INLINING=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 ROM?

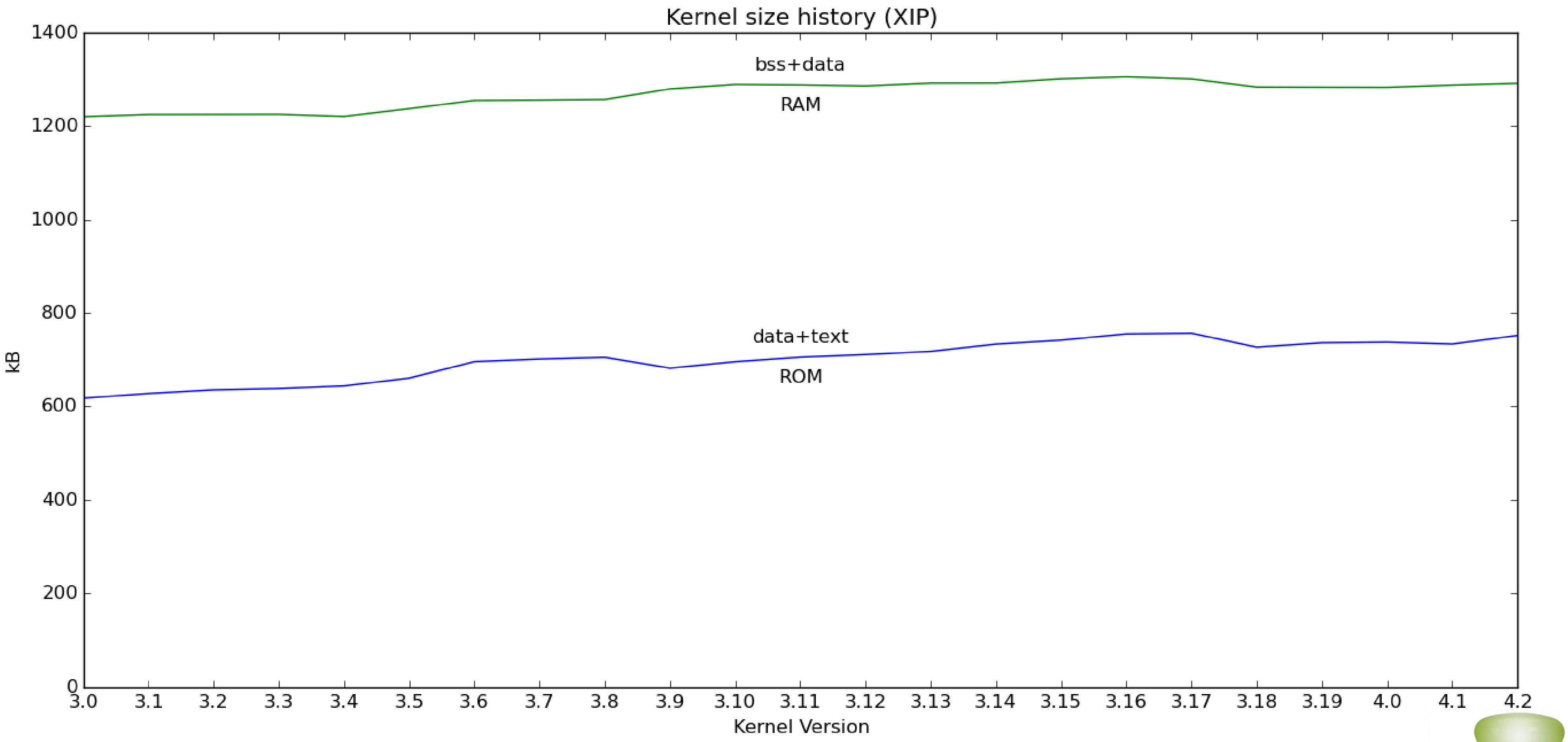
- For XIP (Execute in-place):
  - .text + .data => FLASH
  - .bss + .data => RAM
- For compressed kernel image:
  - bzImage => FLASH
  - .bss + .data + .text => RAM

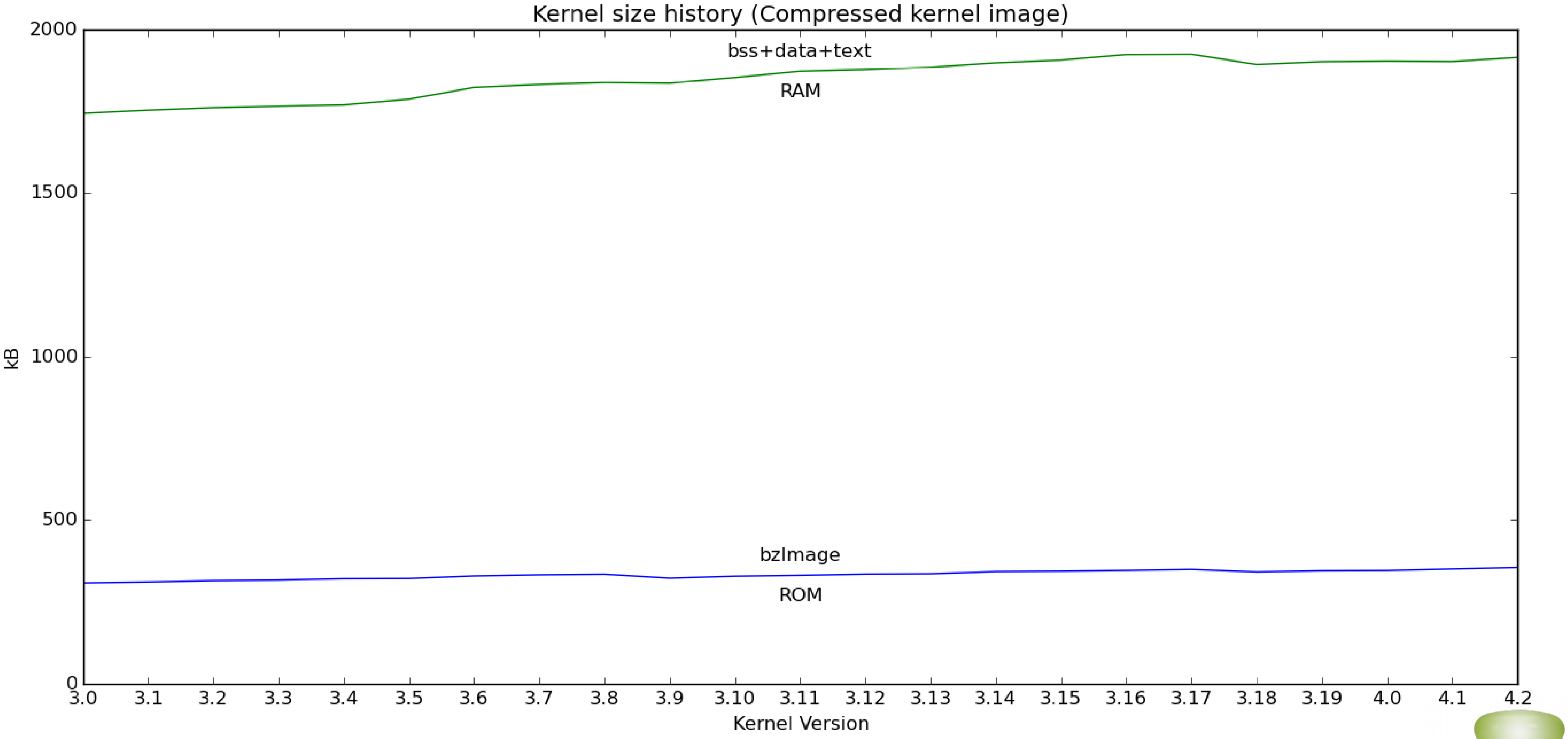
XIP versus Compressed Image

- With XIP:
  - FLASH must be directly addressable by CPU
  - Kernel stored in FLASH (uncompressed)
  - Executes .text from FLASH
  - Bootstrap code copies .data from FLASH to RAM
- Trade-off:
  - Saves RAM at the expense of FLASH

XIP versus Compressed Image

- With Compressed Image:
  - FLASH need not be directly addressable by CPU
  - Entire kernel copied from FLASH to RAM
  - Kernel self-decompresses and executes in RAM
- Trade-off:
  - Saves FLASH at the expense of RAM


Kernel size history (XIP)
bss+data
RAM
data+text
ROM
kB
0
200
400
600
800
1000
1200
1400
3.0
3.1
3.2
3.3
3.4
3.5
3.6
3.7
3.8
3.9
3.10
3.11
3.12
3.13
3.14
3.15
3.16
3.17
3.18
3.19
4.0
4.1
4.2
Kernel Version


Kernel size history (Compressed kernel image)
kB
2000
1500
1000
500
0
bss+data+text
RAM
bzImage
ROM
3.0
3.1
3.2
3.3
3.4
3.5
3.6
3.7
3.8
3.9
3.10
3.11
3.12
3.13
3.14
3.15
3.16
3.17
3.18
3.19
4.0
4.1
4.2
Kernel Version

A brief history: The kernel weight-watchers

- The kernel 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?
- 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
- New tools to hunt for bloat

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

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

Demo time

References

- Linux tiny
  - http://events.linuxfoundation.org/sites/events/files/slides/tiny.pdf
  - https://lwn.net/Articles/608945/
  - http://elinux.org/images/5/5c/Linux-tiny-revival-jamboree16.pdf
  - 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
  - http://elinux.org/index.php?title=Kernel_Size_Tuning_Guide
- Tips
  - https://lwn.net/Articles/370998/

Acknowledgements

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