Neither Seen Nor Heard: An Alternative View of the State of Firmware

R. Hale, V. Zimmer
Firmware Architects
Intel Corp.
June, 2010

Ground Rules

- By firmware, we don't just mean code on a chip
  - With the advent of large NAND flash devices, the old definition is losing its meaning
  - Perhaps "Embedded software" is a better description
- We lump silicon and board engineering into Hardware engineering
- In this presentation we focus on large generational system firmware
  - Large: Say >64KB (Not e.g. 4 bit micros)
  - Generational: Not for one-off products but for successive generations of similar products, say PCs or MP3 players
    - Continuum for software code base, test library
  - System: Not ucode

How did you get to this workshop?

All photographs by the authors

Fly?

Continental Micronesia
ATR-42
(Guam, 3/2010)

Firmware appears in modern aircraft everywhere from the glass cockpit to navigation to the entertainment system. In a 737-500, the avionics alone has ~50 embedded processors*

(*private communication)

Continental Micronesia
Boeing 737-800
(Iwo Jima, 3/2010)

Take the Train?

Firmware is here too, from traction control to performance monitoring to the “black boxes”.

Amtrak
General Electric P42
(Sacramento CA, 2009)

Drive?

You could have driven here without using firmware but it would have been a challenge
(Sienna, Italy, 9/2004)

The *average* modern auto uses 50 microprocessors, each of which has firmware.
(Avignon, France, 9/2004)

Run, Bike?

Upper right: Creative Nomad DAP-TD00002
Upper left: Sony Walkman NWZ-B103F
Lower right: SanDisk Sansa Fuse BH0809AXVK

Without firmware, it was a quiet trip.

MP3 players all have firmware in them.

The digital camera that took these pictures (Panasonic Lumix TZ-5) has firmware in it

Even if you stay home…

- Entertainment
  - The DVD player, the flat screen TV, video games, at least
- Kitchen
  - The microwave, the food processor, the oven front panel (at least)
- Infrastructure
  - Tankless hot water heater, thermostat, etc.
- Home computer
  - BIOS(s), Keyboard, Mouse, Hard Disk and DVD drives, monitor, printer, fan, power supply, printer
- Cell phones

Firmware is everywhere

The Crux of the Matter

- If firmware is so ubiquitous and works so much of the time, why is there a pervasive perception that it is
  - Developed using poor engineering techniques?
  - Difficult to validate?
  - Unreliable?

As many permutations and combinations of causes as there are projects

Notable causes

- Firmware is software so can suffer well known software ills
- Firmware relies on re-use and is tightly bound to hardware, which can vary considerably from generation to generation
- Firmware is most commonly developed in a hardware world to hardware schedules and processes
  - Difficult to exploit software malleability
  - Clash of hardware dev, firmware dev, hardware validation cultures
- Firmware has many masters, all priority 1
  - Including H/W validation
  - H/W validation conservatism at odds with continuous improvement
- Incentives to fix issues in firmware cause root cause to be commonly incorrectly assigned to firmware
- Lack of understanding of how firmware is properly validated

Case study: BIOS

- Basic I/O Subsystem †
  - Reset vector to OS hand off: "Power On Self Test" (POST)
  - Also (small number of) run-time services
- BIOS itself validated mainly as black box
  - BIOS is used during silicon power up, board validation as well as being an integral component in production systems
- Expect >90% reuse from one processor / chipset generation to the next
  - Extremely high relative to other software

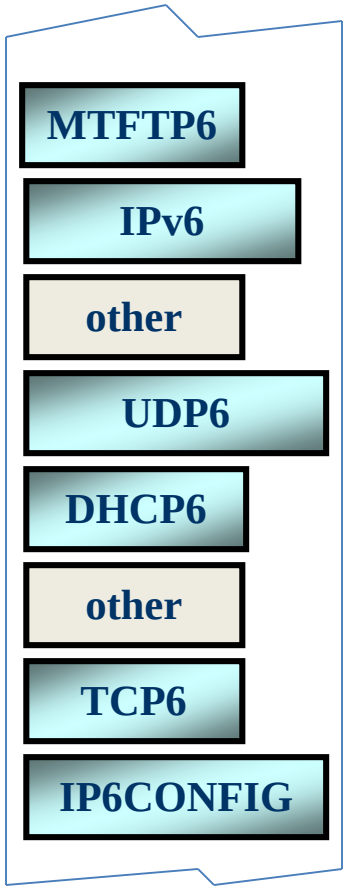
1980s: "Legacy BIOS"	**Today: "UEFI/PI"*** **
< 64KB of assembly code*	~1.4MB of C and some assembly code**
POST more test than initialization	POST almost all initialization: RAM init 100KB
Monolithic	Phases of kernels and communicating modules

†not Blame It On Software

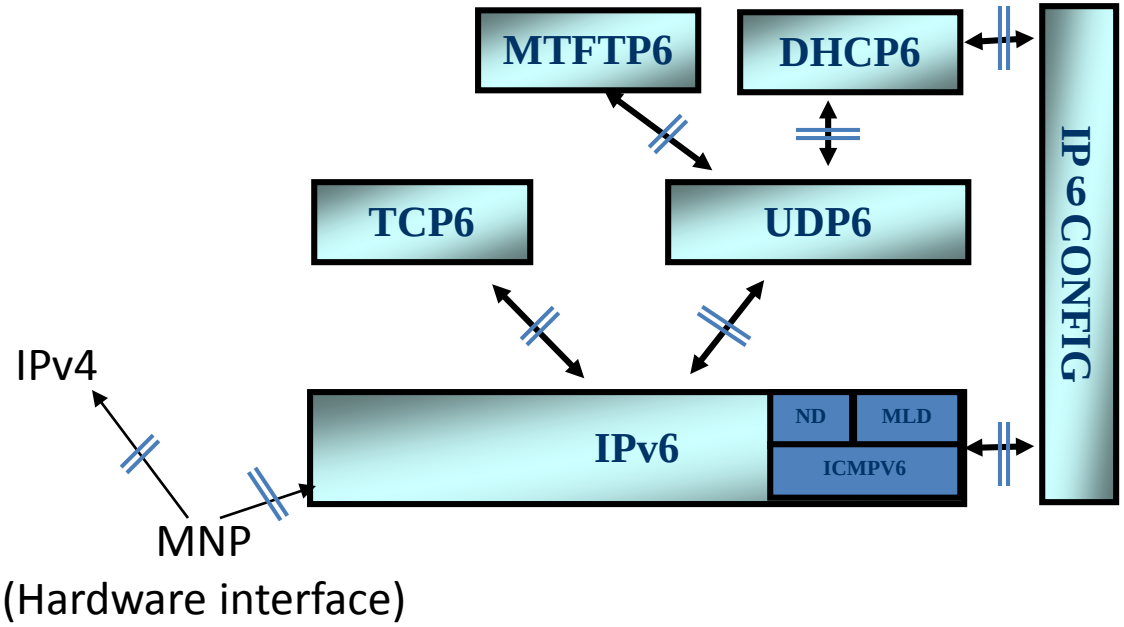
* IBM PC/AT
** Intel® Desktop Board
*** See uefi.org for specs

Modules: The New Black Boxes

In Flash



During Execution



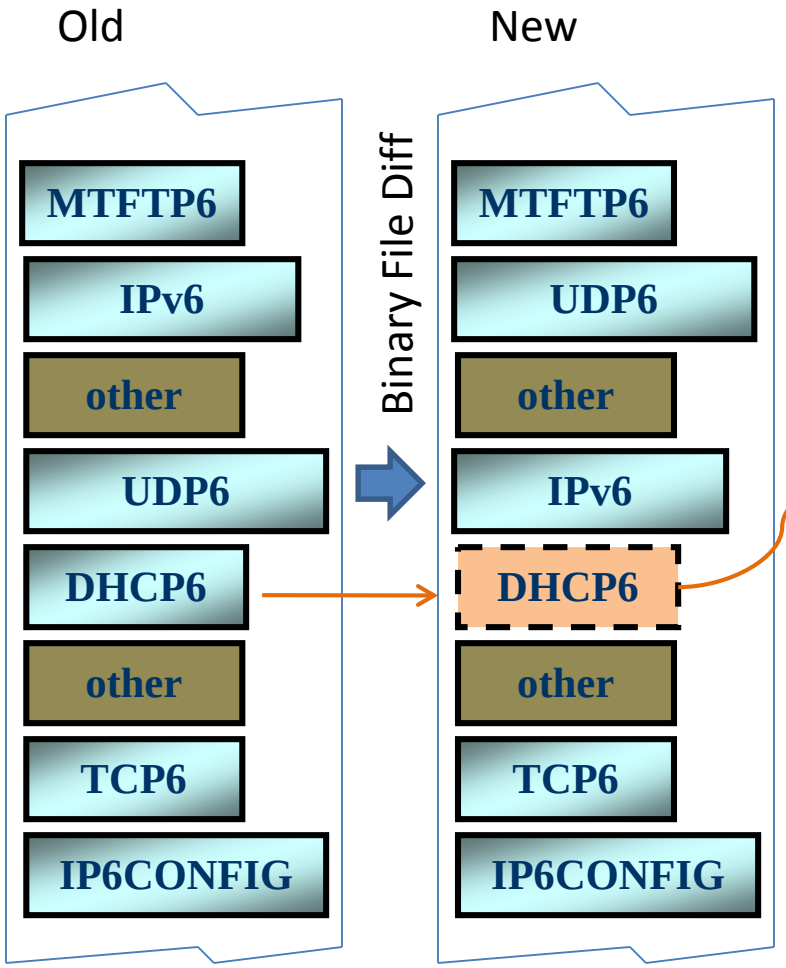
In UEFI, modules are stored in flash like files.
During execution, modules use light-weight communication interfaces to interact. Can now treat each module as a black box , testing each module's interfaces.*

TCP6 = module

= interface

* see e.g. SCTs at uefi.org

Focusing Coverage



Module / Test Database

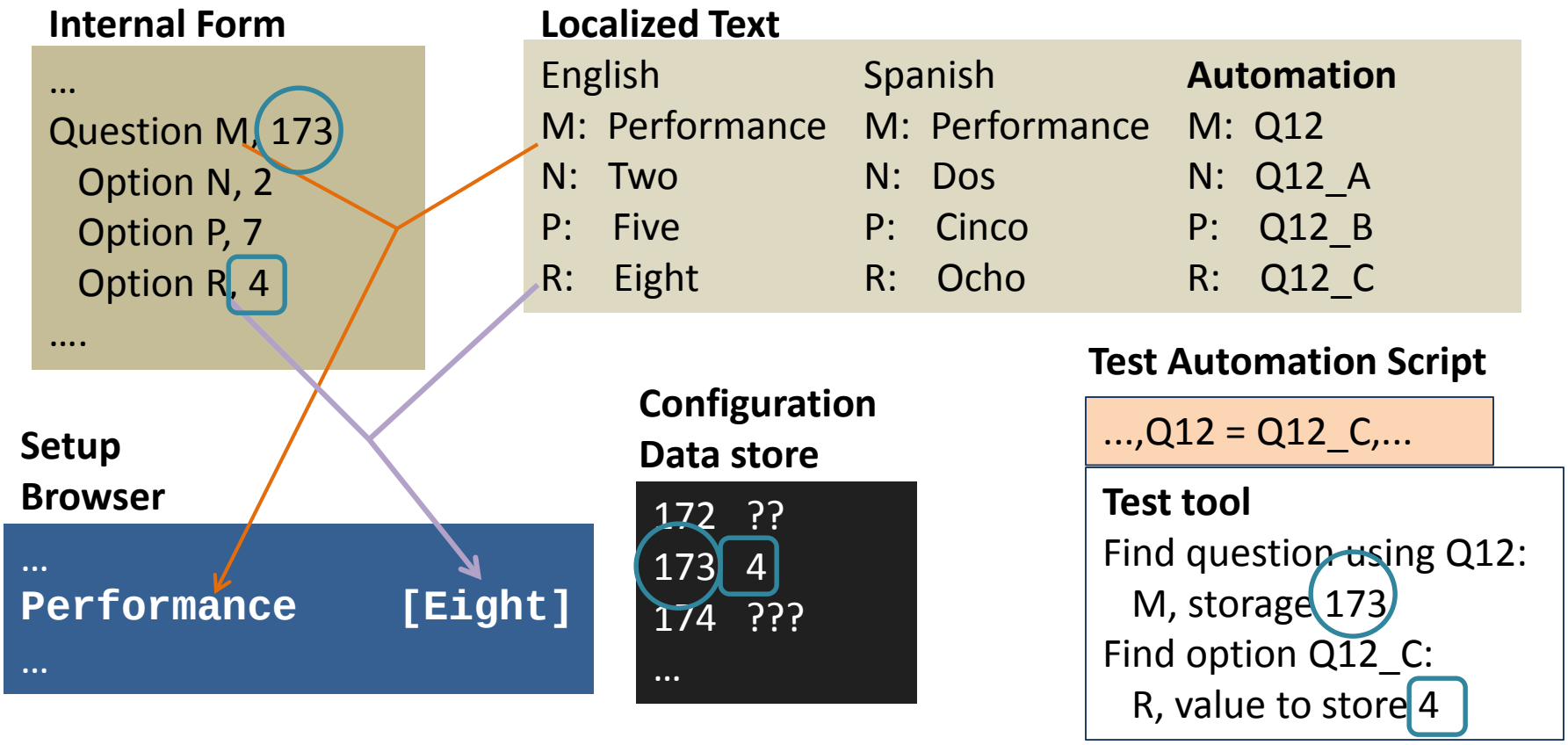
Module	Tests
MTFTP6	1,2,3,4,8,23
UDP6	7,8,12,15
IPV6	3,5,7, 17, 18
DHCP6	9,10,11,18,22
…	…

Rather than 23 tests, we run 5.
We augment coverage with fixed set of go / no-go tests to account for cross module interactions

Issue: Stale database

Module-level testing reduces test time, increases coverage

Reconfiguration Automation in UEFI



UEFI leaves all forms data at OS hand off. By adding an "automation language", the same forms that Setup uses can be used to perform automatic reconfiguration. The tools use a script instead of a user but are otherwise similar to simple browsers.

Cooperation, not distrust

- The result of these and other improvements in BIOS evaluation?
  - The state of the art does more with less
  - Industry going lean
  - Complexity increasing

  => Running as fast as we can just to stay where we are
- The next tool in the chest starts with the realization that we are developing full products
  - Requires cooperation among disciplines
  - Training in all disciplines starting in college

Without the wind, the grass does not move.
Without software, hardware is useless.

—Geoffrey James