



Growing the SCMI support on FreeBSD

Cristian Marussi – Staff Software Engineer @ ARM
cristian.marussi@arm.com

Agenda

- + whoami (... with a “disclaimer”)
- + What’s the problem(s) ... why SCMI ?
- + SCMI Overview
- + Deployments Scenarios
- + SCMI Messaging
- + SCMI FastChannels
- + Coexistence on ACPI systems
- + Linux SCMI Stack (as a reference example)
- + Current SCMI Support in FreeBSD
- + Work in progress
- + ... Next

What's the problem ?

- + Modern SoC are complex beasts ...
 - ... composed by a number of **different logic elements** providing a number of functionalities
 - ... but, usually, **not all together** at the same time and **not at the same level** of performance
- + ... a well-known design **strategy to maximize efficiency** is to have it:
 - ... **partitioned** in a number of distinct power/voltage **domain islands**
 - ... so that such islands can be **selectively configured** based on actual runtime needs
- + ... such islands will definitely:
 - ... have different on-chip users requiring, dynamically, **different configurations**
 - ... with such users frequently using **different SW stacks** with different security needs
- + ... such users, potentially conflicting, configuration **requests will be served**
 - ... by some sort of **central entity** who is in charge of policing

... BUT ...

I - Why SCMI ?

- + We don't have a **common language** to express such requests so
=> a number of protocols have come into existence in the recent past:
TI/SCI, QCOM/RPM, Nvidia/BPMP, ARM/SCPI (!)

SCMI - System Control and Management Interface [1]

→ aims to unify this with a new **standard common protocol** abstraction ...

“... System Control and Management Interface (SCMI), which is a set of operating system-independent software interfaces that are used in system management.”

... in these regards, of course, this could come to mind ...

II - Why SCMI ? - <https://xkcd.com/927/>

HOW STANDARDS PROLIFERATE:
(SEE: A/C CHARGERS, CHARACTER ENCODINGS, INSTANT MESSAGING, ETC.)



III - Why SCMI ?

... BUT ...

since SCMI is more **flexible**, easily **extensible** and transport independent and ...
... thanks to the efforts of Arm, Linaro and our partners ...

→ an **increasing number of real-world systems have adopted SCMI** recently...

- ... in a number of different **segments**: Mobile / Automotive / Embedded / PCs
- ... even amongst the vendors that had previously developed their own protocol

... so this gives us some sort of **validation** ...

+ ... and I am here pestering you about this SCMI thing :P

SCMI Overview - I

+ Based on a **client-server model**

- one SCMI server, called **platform**, which can live in a number of places
- multiple SCMI clients, called **agents**, issuing possibly conflicting requests to the server
- multiple possible transports are possible (in Linux MBOX/SMC/VIRTIO/FFA/OPTEE) [out-of-spec]
- each agent has one or more dedicated communication channel
 - agents are identified by the server from the channel they speak from

+ **OS agnostic** specification provides protocol interfaces dedicated to system management

+ **Fully discoverable** around features, services and available resources (except transport characteristics)

- a mandatory Base protocol to discover the list of implemented protocols (and other things...)

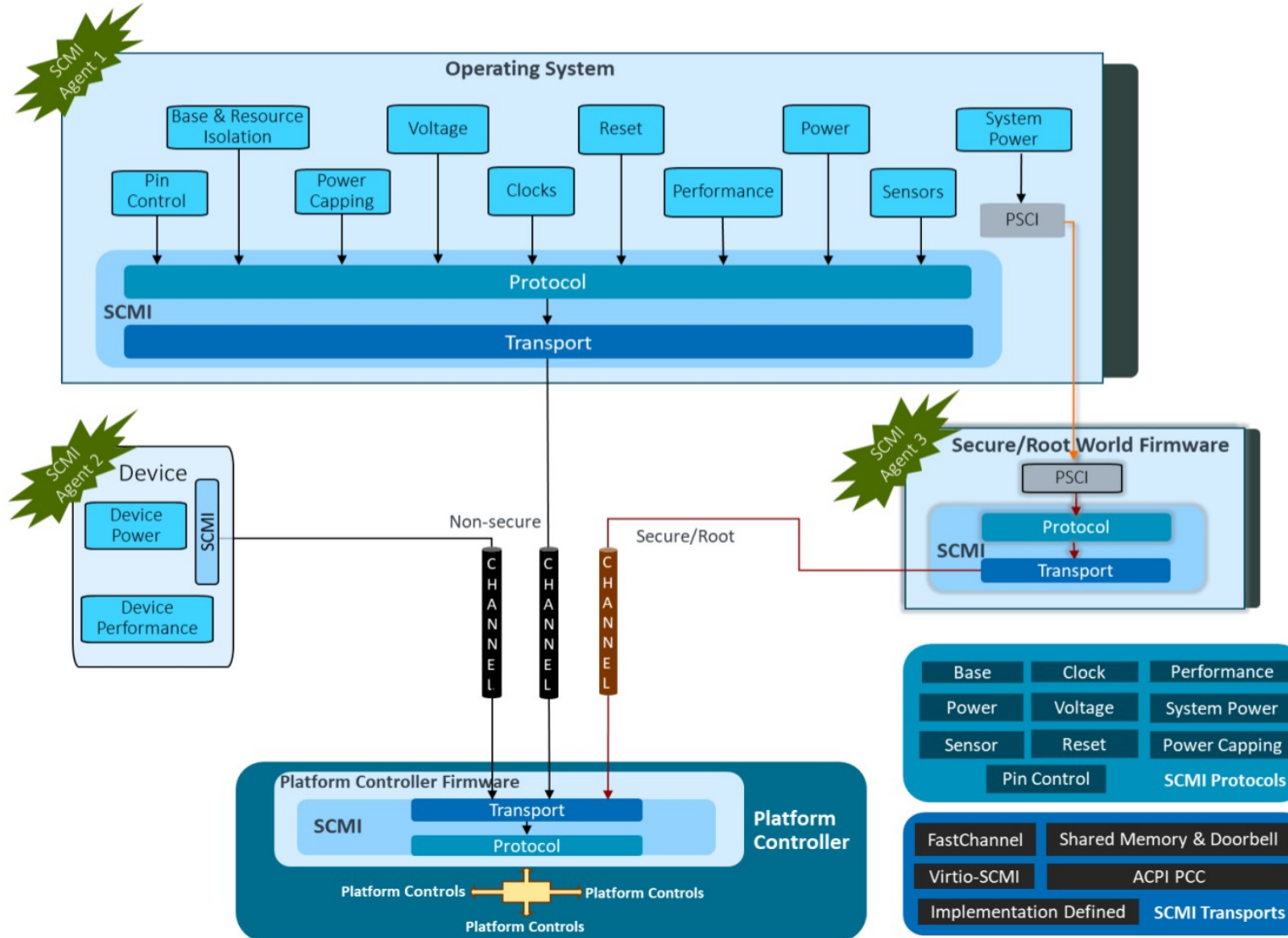
+ **Extensible**

- per-domain dedicated sub-protocols (voltage, power, performance) ... external contributions possible
- optional vendor protocols

SCMI Overview - II

- + **Abstraction** and unification of accesses to managed resources
 - no need to have hw specific drivers kernel-side
 - ...as long as you have them platform side ... and you speak SCMI (not necessarily for everything)
- + **Policy enforcing** on a per-client basis, SCMI server is the ultimate arbiter:
 - can deny, silently or not, any agents' request based on server configuration
- + **Enhanced security** by delegation of policing and actions to a **smaller TCB living in a secure world**
 - CLKscrew <https://www.youtube.com/watch?v=0tM2v2SZDxY> ... server can deny unsafe requests even from rooted OS
- + **Virtualization friendly** design
 - server can expose a different **views** of the system to each agent (VMs)
 - same shared resource, different ID (ex. Clock) **OR** different resource, same ID (ex. sensor)
 - **harmonization** of potentially conflicting requests around shared resources
- + transport independent...not part of the specification really....

SCMI Overview - III



SCMI Server Deployment Scenarios

Having a number of transports available means the SCMI server can be **deployed in a number of different places**:

- in a dedicated MCU, most probably in Secure world
- in a TrustedApplication which run within the context of a TEE OS (like OPTEE)
- in a SecurePartition running at S-EL2, S-EL1 or S-EL0 depending on arch support
- in a dedicated Virtual Machine
- embedded in EL3 TF-A (discouraged...)

A **reference SCMI Server** platform is implemented in **ARM SCP Firmware** [1], and a compliance test suite is available too [2] ... but vendors can write **their own** servers ...

[1] <https://gitlab.arm.com/firmware/SCP-firmware>

[2] <https://gitlab.arm.com/tests/scmi-tests>

SCMI Messaging - I

Agents/Platform communications use 2 kinds of channels:

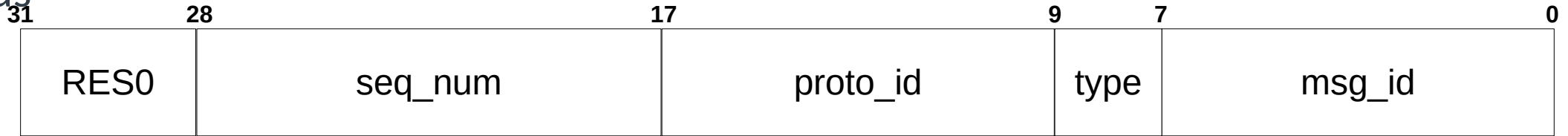
- **A2P**: agent to platform channels carrying the **agent initiated** transactions, i.e. command issued by the agent and the immediate responses from the platform
- **P2A** (optional): platform to agent channels carrying the **platform initiated** transactions, i.e. delayed responses and notifications

... and 2 kinds of commands:

- **Synchronous Commands** - agent issues a command on A2P and the response is immediately delivered on that same channel, which is *kept busy* until the request has been served and the reply received.
- **Asynchronous Commands** - agent issues a command on A2P and an immediate status-only response is delivered on that same channel: such immediate reply does NOT carry any effective payload but ONLY the status, so that the channel is *freed immediately*; the full-response will be delivered **later** (once computed) using a **delayed response** sent on **P2A**

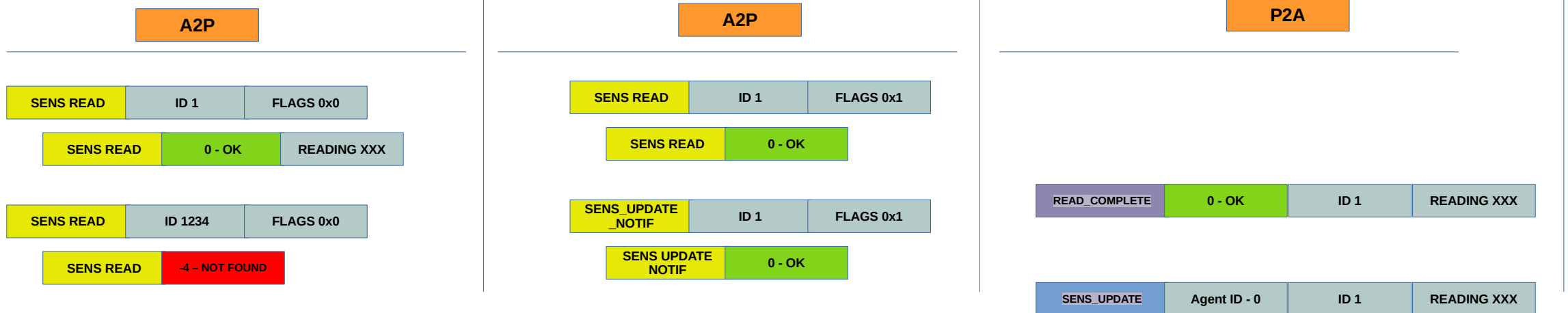
SCMI Messaging - II

Each message is composed by a 32-bit **header** and an optional payload of <N> 32-bit words



Types :: 0 – command/response 2 – delayed response 3 – notification

A response message is built by returning **exactly the same header**, unchanged, plus any optional payload. (except for delayed response where the type is changed). Policy around the usage of the **seq_num** field is completely in the hand of the agent.



SCMI FastChannels

Beside the standard messaging channels, based on a command-reply pattern, FastChannels are provided as an **alternative** messaging mechanism but only for:

- some specific commands in a few protocols
- a specific resource

“A FastChannel is a lightweight unidirectional channel that is dedicated to a single SCMI message type for controlling a specific platform resource.”

In a nutshell the **server can advertise** (via regular messaging) some well defined memory **areas** where the agent can **read/write directly the payload** for a well defined command related to a well defined resource, avoiding the command-reply overhead.

SCMI on ACPI Systems

Most of the SCMI Agent support currently needed in Linux/FreeBSD is targeted at DeviceTree based systems...which is where most of the work is needed

ACPI-based implementations **can leverage SCMI protocols** to provide platform services using standard ACPI methods (SCMI specification is kept ACPI compatible by ATG)

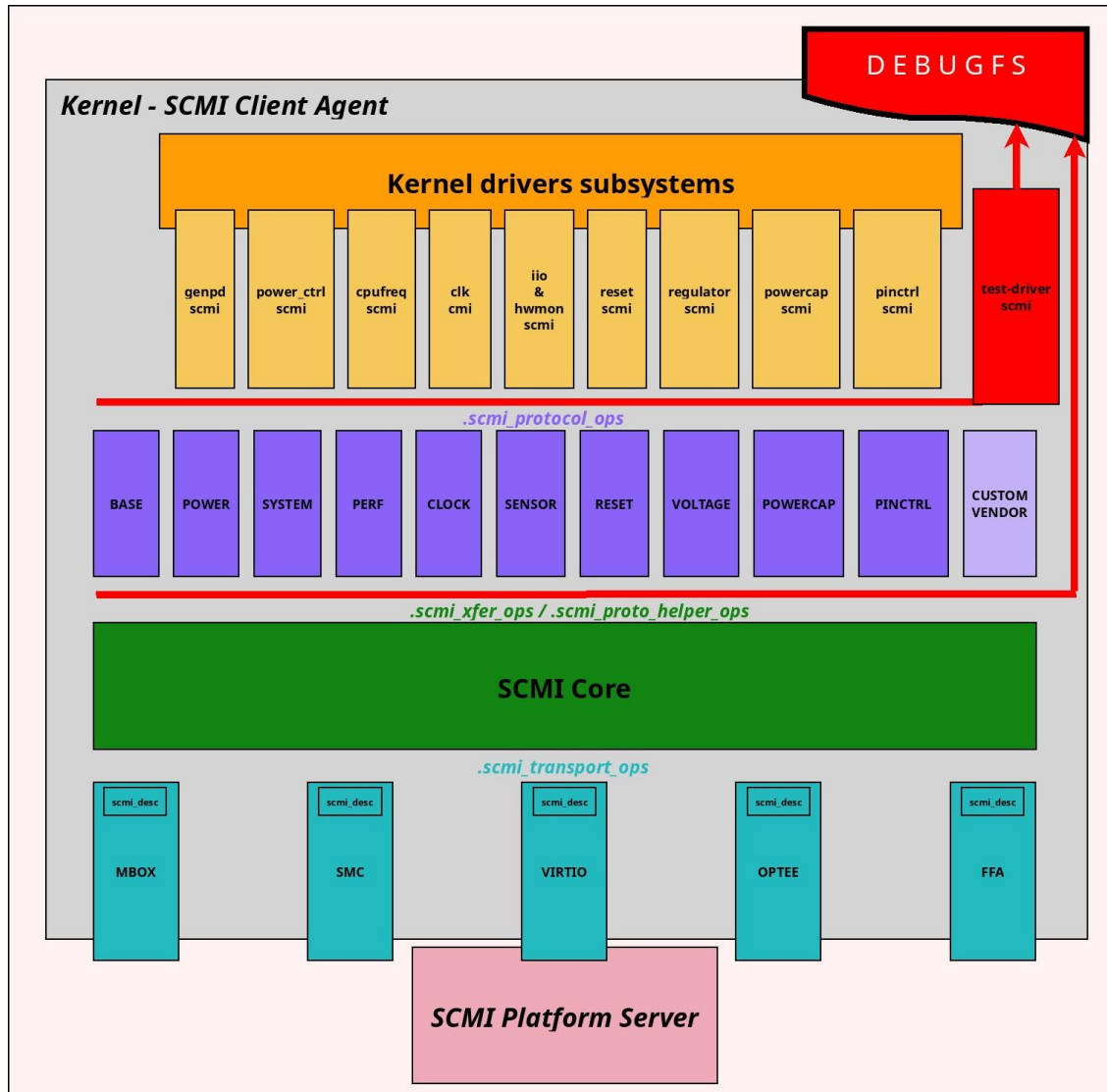
As an example, a device may be power managed by ACPI-aware OS using the **standard ACPI control** methods.

These ACPI methods can **send SCMI Power Management Protocol requests** to the platform to transition the power state of the device.

SCMI **MBOX/SHMEM transport** channels can be represented as an ACPI Platform Communications Channel (**PCC**) of **Type 3**.

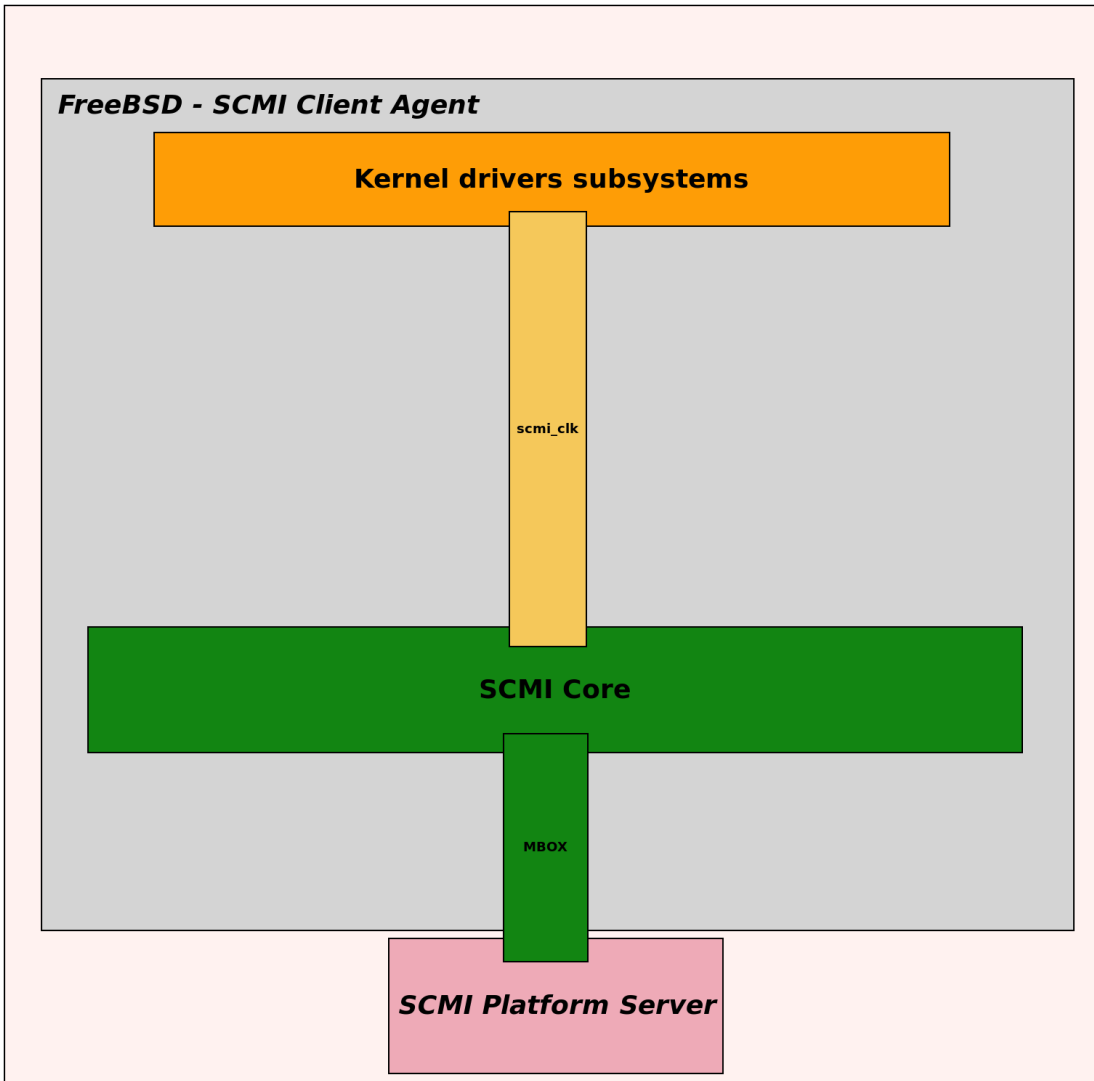
The server on the other end does NOT really know if the request originated from a DT or an ACPI system...as long as you have some **glue** code in the ACPI stack (TBD)

Linux SCMI Stack



- **Layered design**
 - **transport:**
 - Abstract medium specific details
 - MBOX/SMC/OPTEE use a well defined SharedMemory area to carry the message
 - **core:**
 - Message tracking (*seq_num*)
 - Handles replies, timeout, errors, late replies
 - **protocols:**
 - Knows how to build the messages for a specific task
 - SCMI driver **users:**
 - Plug into various kernel subsystems (clocks)
 - Call into protocol ops (*clock_enable*)
- 2 test and development facilities in DEBUGFS
 - **SCMI Raw** - inject/snoop messages to:
 - test the Server with the compliance suite
 - test the basic core messaging functionalities
 - **scmi-test-driver** – invoke *scmi_ops* to:
 - test the SCMI protocol APIs (not upstream)

FreeBSD SCMI Stack - I



Initial FreeBSD SCMI Support

- Initial commit in 2022

commit 54b96380f5774c1754a0fcf25212fa8e01db74f6

Author: Ruslan Bukin <br@FreeBSD.org>

Date: Mon Dec 19 20:16:18 2022 +0000

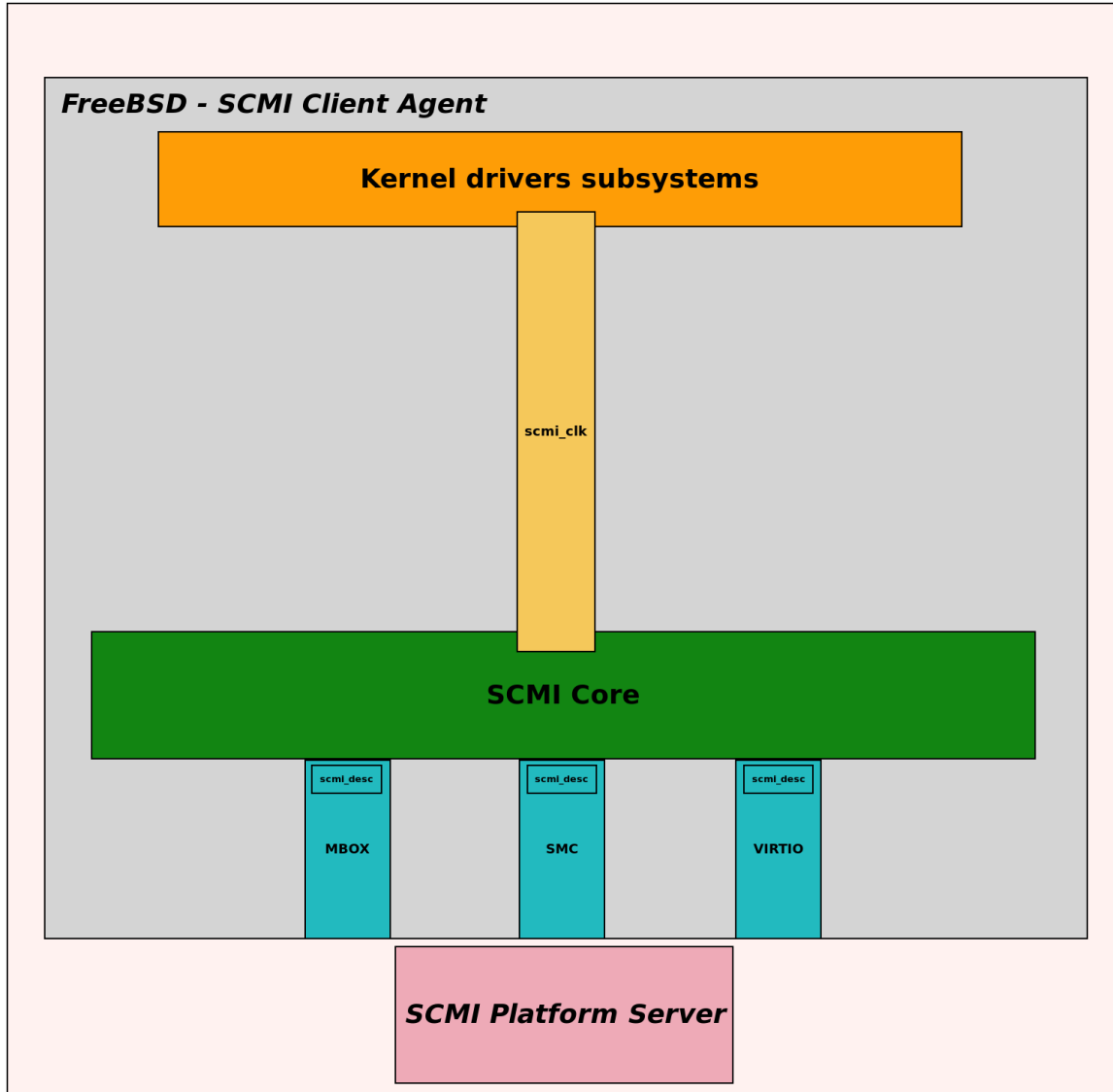
Add support for ARM System Control and Management Interface (SCMI) v3.1.

- added minimal support for
 - 1 transport (MBOX)
 - “Basic” core facilities (only sync-commands)
 - SCMI clocks
- ... added support was **minimal for a very good reason:**
 - **ONLY 1 SCMI platform** (server) available on FreeBSD
 - Morello implementing only Clock protocol
- ...today → **not so many more** FreeBSD/SCMI platforms

BUT...

- the Server **can now be deployed in a number of places**
 - including in some VM or in Userspace

FreeBSD SCMI Stack - II



Current FreeBSD SCMI Support

...so the **first addition from Arm** around mid-2024 was:

- Splitting out the transport layer and adding:
 - **VirtIO transport** [1]
 - SMC transport
- Restructure a bit of the SCMI Core in preparation of more complex messaging support (still only sync-cmd...)

Since enabling the use of a **virtualized SCMI Server**:

→ allows for more complex SCMI development on FreeBSD SCMI Agent capabilities **without real HW** available

→ allows anyway the FreeBSD SCMI Agent to be used in an existing **virtualized deployment**

[1] <https://docs.oasis-open.org/virtio/virtio/v1.2/csd01/virtio-v1.2-csd01.html#x1-60400017>

FreeBSD - Guest SCMI Agent with KVMtool/SCMI_EMU

```
Copyright (c) 1979, 1980, 1983, 1986, 1988, 1989, 1991, 1992, 1993, 1994
The Regents of the University of California. All rights reserved.
FreeBSD is a registered trademark of The FreeBSD Foundation.
FreeBSD 15.0-CURRENT #1 scmi_dev-n72002-3fcd791f8463: Tue Sep 3 17:35:11 BST 2024
cris@pluto:/home/cris/dev/src/freebsd/projects/kvmtool_guest/src/arm64.aarch64/sys/GENERIC_KVMT00L_arm64
Clang version 17.0.6 (Fedora 17.0.6-2.fc39)
WARNING: WITNESS option enabled, expect reduced performance.
VT: init without driver.
module scmi already present!
real memory = 2147483648 (2048 MB)
avail memory = 2667902464 (1972 MB)
Starting CPU 1 (1)
FreeBSD/SMP: Multiprocessor System Detected: 2 CPUs
arc4random: WARNING: initial seeding bypassed the cryptographic random device because it was not yet seeded and the knob 'bypass_before_seeding' was enabled.
Random: entropy device external interface
kdb0 at kbdmux0
ofwbus0: <Open Firmware Device Tree>
regfix0: <Fixed Regulator> on ofwbus0
regfix1: <Fixed Regulator> on ofwbus0
ofw_firmware0: <OFW Firmware Group> on ofwbus0
psc0: <ARM Power State Co-ordination Interface Driver> on ofwbus0
gic0: <ARM Generic Interrupt Controller v3.0> mem 0x3fff0000-0x3ffff000-0x3ffff000 on ofwbus0
its0: <ARM GIC Interrupt Translation Service> mem 0x3fff9000-0x3ffff000 on gic0
generic_timer0: <ARMv8 Generic Timer> irq 0,1,2,3 on ofwbus0
Timecounter "ARM MPCore Timecounter" frequency 24000000 Hz quality 1000
Event timer "ARM MPCore Eventtimer" frequency 24000000 Hz quality 1000
cpu1st0: <Open Firmware CPU Group> on ofwbus0
cpu0: <Open Firmware CPU> on cpu1st0
ns8250: UART FCR is broken
ns8250: UART FCR is broken
ns8250: UART FCR is broken
uart0: <Non-standard ns8250 class UART with FIFOs> mem 0x1000000-0x1000007 irq 4 on ofwbus0
uart0: console (-1,n,8,1)
ns8250: UART FCR is broken
ns8250: UART FCR is broken
ns8250: UART FCR is broken
uart1: <Non-standard ns8250 class UART with FIFOs> mem 0x1001000-0x1001007 irq 5 on ofwbus0
ns8250: UART FCR is broken
ns8250: UART FCR is broken
ns8250: UART FCR is broken
uart2: <Non-standard ns8250 class UART with FIFOs> mem 0x1002000-0x1002007 irq 6 on ofwbus0
ns8250: UART FCR is broken
ns8250: UART FCR is broken
ns8250: UART FCR is broken
uart3: <Non-standard ns8250 class UART with FIFOs> mem 0x1003000-0x1003007 irq 7 on ofwbus0
pci0: <Generic PCI host controller> mem 0x40000000-0x4fffffff on ofwbus0
pci0: <OFW PCI bus> on pci0
virtio_pci0: <VirtIO PCI (modern) SCMI adapter> port 0x6200-0x62ff mem 0x50000000-0x500000ff,0x50000040-0x500007ff irq 8 at device 0.0 on pci0
Vtscm0: <VirtIO SCMI Adapter> on virtio_pci0
Vtscm0: Platform supports P2A channel.
virtio_pci1: <VirtIO PCI (modern) Network adapter> port 0x6300-0x63ff mem 0x50000080-0x500000ff,0x50000c00-0x50000fff irq 9 at device 1.0 on pci0
Vtnet0: <VirtIO Networking Adapter> on virtio_pci1
Vtnet0: Ethernet address: 02:15:15:15:15:15
Vtnet0: netmap queues/slots: TX 1/256, RX 1/128
000 000007 [ 452] vnet_netmap_attach vnet attached txq-1, txd-256 rxq-1, rxd-128
virtio_pci2: <VirtIO PCI (modern) block adapter> port 0x6400-0x64ff mem 0x50001000-0x500010ff,0x50001400-0x500017ff irq 10 at device 2.0 on pci0
vtblk0: <VirtIO Block Adapter> on virtio_pci2
vtblk0: 6176MB (12649501 512 byte sectors)
scmi_virtio0: <ARM SCMI VirtIO Transport driver> on ofw_firmware0
scmi_virtio0: Fed 32 initial P2A buffers to platform.
Vtscm0: Enabled interrupts on VQ[0].
scmi_virtio0: VirtIO cmdq virtqueue configured - cmdq_sz:16
Vtscm0: Enabled interrupts on VQ[1].
scmi_virtio0: VirtIO evq virtqueue configured - evq_sz:32
scmi_virtio0: Transport reply timeout initialized to 100ms
scmi_clk0: <SCMI Clock Management Unit> on scmi_virtio0
scmi_clk0: Found 10 clocks.
scmi_clk0: Clock 'emu0_clk_fixed' registered.
scmi_clk0: Clock 'emu1_clk_4_rts' registered.
scmi_clk0: Clock 'emu2_clk_segmented_longggg_name' registered.
scmi_clk0: Clock 'emu3_clk_segmented_fully_restricted' registered.
scmi_clk0: Clock 'emu4_clk_fixed_parent_AAA' registered.
scmi_clk0: Clock 'emu5_clk_fixed_parent_BBB' registered.
scmi_clk0: Clock 'emu6_clk_fixed_parent_CCC' registered.
scmi_clk0: Clock 'emu7_clk_fixed_parent_DDD' registered.
scmi_clk0: Clock 'emu8_clk_fixed_child_AB_rate_restricted' registered.
scmi_clk0: Clock 'emu9_clk_fixed_child_CD_state_restricted' registered.
armv8crypto0: <AES-CBC,AES-XTS,AES-GCM>
Timecounters tick every 1.000 msec
18:38 $ ./scmi_emu -s /tmp/
SCMI Emulation v0.1-beta
Copyright (c) 2024, Arm Limited or its affiliates. All rights reserved.
Registered protocol 0x10
Registered protocol 0x11
Registered protocol 0x12
Registered protocol 0x13
Registered protocol 0x14
Registered protocol 0x15
Registered protocol 0x16
Registered protocol 0x17
Registered protocol 0x18
Registered protocol 0x19
Registered protocol 0xFF
Sockets configured.
RX Timeout 3000ms
SYSPower - timeout:10000 flags:0x1
Waiting for APZ connection on /tmp/scmi_ospm_vq0.sock ...OK !
Waiting 10 secs for optional P2A connection on /tmp/scmi_ospm_vq1.sock ...OK !
Starting Notifications thread...
Spawning RX threads...waiting for workers.....32 OK !
Entering packet loop...
1726226429.606404:==>PLAT[0]:[0x14]::CMND[0x1]::[0000] -- [0/REQ_0000]
1726226429.606500: <=>PLAT[0]:[0x14]::RESP[0x1]::[0000] -- [00000000 4/REQ_0000]
1726226429.608652:==>PLAT[0]:[0x14]::CMND[0x3]::[0001] -- [00000000 4/REQ_0000]
1726226429.608734: <=>PLAT[0]:[0x14]::RESP[0x3]::[0001] -- [00000000 08000000 30756065 686C635F 7869665F 0006465 24/REQ_0000]
1726226429.610991:==>PLAT[0]:[0x14]::CMND[0x3]::[0002] -- [00000001 4/REQ_0000]
1726226429.611075: <=>PLAT[0]:[0x14]::RESP[0x3]::[0002] -- [00000000 08000000 31756065 686C635F 725F345F 00007374 24/REQ_0000]
1726226429.613312:==>PLAT[0]:[0x14]::CMND[0x3]::[0003] -- [00000002 4/REQ_0000]
1726226429.613398: <=>PLAT[0]:[0x14]::RESP[0x3]::[0003] -- [00000000 68000000 32756065 686C635F 6765735F 00657460 24/REQ_0000]
1726226429.615397:==>PLAT[0]:[0x14]::CMND[0x8]::[0004] -- [00000002 4/REQ_0000]
1726226429.615526: <=>PLAT[0]:[0x14]::RESP[0x8]::[0004] -- [00000000 00000000 32756065 686C635F 6765735F 6E657460 5F646574 676E6F6C 5F676767 6560616E 00000 000 0000000 00000000 00000000 00000000 00000000 68/REQ_0000]
1726226429.617817:==>PLAT[0]:[0x14]::CMND[0x3]::[0005] -- [00000003 4/REQ_0000]
1726226429.617906: <=>PLAT[0]:[0x14]::RESP[0x3]::[0005] -- [00000000 68000003 33756065 686C635F 6765735F 006E6560 24/REQ_0000]
1726226429.619953:==>PLAT[0]:[0x14]::CMND[0x3]::[0006] -- [00000003 4/REQ_0000]
1726226429.620071: <=>PLAT[0]:[0x14]::RESP[0x3]::[0006] -- [00000000 00000000 33756065 686C635F 6765735F 746E6560 665F6465 796C6C75 7365725F 63697274 0664 574 08000000 00000000 00000000 00000000 00000000 00000000 68/REQ_0000]
1726226429.622349:==>PLAT[0]:[0x14]::CMND[0x3]::[0007] -- [00000004 4/REQ_0000]
1726226429.622418: <=>PLAT[0]:[0x14]::RESP[0x3]::[0007] -- [00000000 20000001 34756065 686C635F 7869665F 005F6465 24/REQ_0000]
1726226429.624454:==>PLAT[0]:[0x14]::CMND[0x8]::[0008] -- [00000004 4/REQ_0000]
1726226429.624540: <=>PLAT[0]:[0x14]::RESP[0x8]::[0008] -- [00000000 00000000 34756065 686C635F 7869665F 705F6465 6E657261 41415F74 00000041 00000000 00000 000 0000000 00000000 00000000 00000000 00000000 68/REQ_0000]
1726226429.626840:==>PLAT[0]:[0x14]::CMND[0x3]::[0009] -- [00000005 4/REQ_0000]
1726226429.626912: <=>PLAT[0]:[0x14]::RESP[0x3]::[0009] -- [00000000 20000001 35756065 686C635F 7869665F 005F6465 24/REQ_0000]
1726226429.628918:==>PLAT[0]:[0x14]::CMND[0x8]::[000A] -- [00000005 4/REQ_0000]
1726226429.629010: <=>PLAT[0]:[0x14]::RESP[0x8]::[000A] -- [00000000 00000000 35756065 686C635F 7869665F 705F6465 6E657261 42425F74 00000042 00000000 00000 000 0000000 00000000 00000000 00000000 00000000 68/REQ_0000]
1726226429.631306:==>PLAT[0]:[0x14]::CMND[0x3]::[000B] -- [00000006 4/REQ_0000]
1726226429.631391: <=>PLAT[0]:[0x14]::RESP[0x3]::[000B] -- [00000000 20000001 36756065 686C635F 7869665F 005F6465 24/REQ_0000]
1726226429.633390:==>PLAT[0]:[0x14]::CMND[0x8]::[000C] -- [00000006 4/REQ_0000]
1726226429.633456: <=>PLAT[0]:[0x14]::RESP[0x8]::[000C] -- [00000000 00000000 36756065 686C635F 7869665F 705F6465 6E657261 43435F74 00000043 00000000 00000 000 0000000 00000000 00000000 00000000 00000000 68/REQ_0000]
1726226429.635935:==>PLAT[0]:[0x14]::CMND[0x3]::[000D] -- [00000007 4/REQ_0000]
1726226429.635997: <=>PLAT[0]:[0x14]::RESP[0x3]::[000D] -- [00000000 20000001 37756065 686C635F 7869665F 005F6465 24/REQ_0000]
1726226429.637885:==>PLAT[0]:[0x14]::CMND[0x8]::[000E] -- [00000007 4/REQ_0000]
1726226429.637960: <=>PLAT[0]:[0x14]::RESP[0x8]::[000E] -- [00000000 00000000 37756065 686C635F 7869665F 705F6465 6E657261 44445F74 00000044 00000000 00000 000 0000000 00000000 00000000 00000000 00000000 68/REQ_0000]
1726226429.642366:==>PLAT[0]:[0x14]::CMND[0x3]::[0010] -- [00000008 4/REQ_0000]
1726226429.642457: <=>PLAT[0]:[0x14]::RESP[0x8]::[0010] -- [00000000 00000000 38756065 686C635F 7869665F 635F6465 646C6968 5F42415F 65746172 7365725F 63697 274 00646574 00000000 00000000 00000000 00000000 00000000 68/REQ_0000]
1726226429.644868:==>PLAT[0]:[0x14]::CMND[0x3]::[0011] -- [00000009 4/REQ_0000]
1726226429.644954: <=>PLAT[0]:[0x14]::RESP[0x3]::[0011] -- [00000000 30000003 39756065 686C635F 7869665F 005F6465 24/REQ_0000]
1726226429.646954:==>PLAT[0]:[0x14]::CMND[0x8]::[0012] -- [00000009 4/REQ_0000]
1726226429.647061: <=>PLAT[0]:[0x14]::RESP[0x8]::[0012] -- [00000000 00000000 39756065 686C635F 7869665F 635F6465 646C6968 5F44435F 74617473 65725F65 69727 473 64657463 00000000 00000000 00000000 00000000 00000000 68/REQ_0000]
```

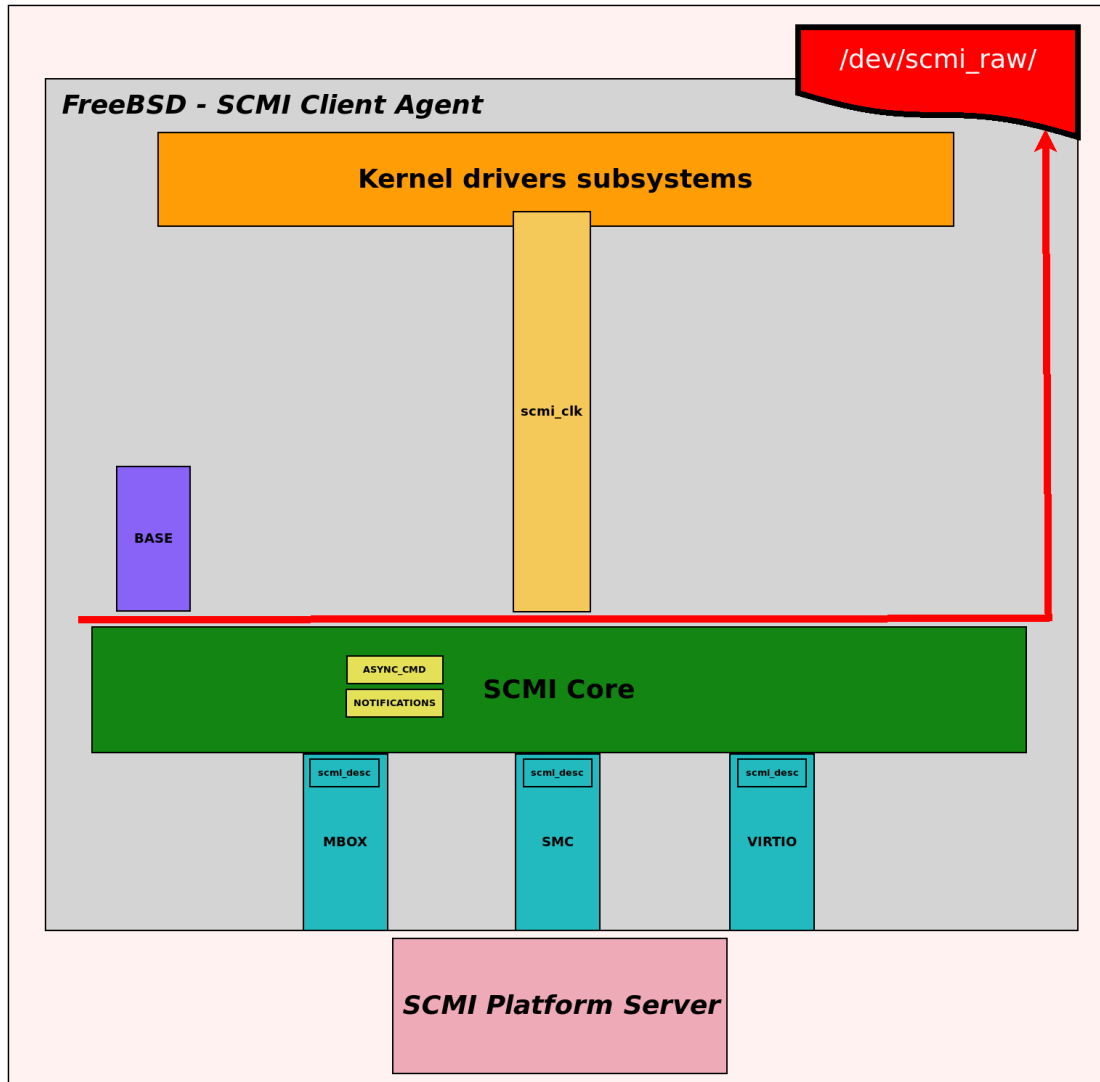


FreeBSD: Guest SCMI Agent with QEMU/SCP

```
Copyright (c) 1992-2023 The FreeBSD Project.
Copyright (c) 1979, 1989, 1993, 1998, 1999, 2000, 2001, 2002, 1993, 1994
    The Regents of the University of California. All rights reserved.
FreeBSD is a registered trademark of The FreeBSD Foundation.
FreeBSD 15.0-CURRENT #146 scmi_dev-n266768-30167ababaa0: Sun Dec  3 22:31:36 GMT 2023
    kris@pluto:opt/freebsd/projects/kvmtool_guest/obj/opt/freebsd/projects/kvmtool_guest/src/arm64.aarch64/sys/GENERIC_KVMT00L_arm64
clang version 16.0.6
WARNING: WITNESS option enabled, expect reduced performance.
VT: init without driver.
module scmi already present!
real memory = 1073741824 (1024 MB)
avail memory = 1023180800 (975 MB)
Starting CPU 1 (1)
FreeBSD/SMP: Multiprocessor System Detected: 2 CPUs
arc4random: WARNING: initial seeding bypassed the cryptographic random device because it was not yet seeded and the knob 'bypass_before_seeding' was enabled.
random: entropy device external interface
kbd0 at kbdmux0
ofwbus0: <Open Firmware Device Tree>
simplebus0: <Flattened device tree simple bus> on ofwbus0
clk_fixed0: <Fixed clock> on ofwbus0
ofw_firmware0: <OFW Firmware Group> on ofwbus0
psci0: <ARM Power State Co-ordination Interface Driver> on ofwbus0
gic0: <ARM Generic Interrupt Controller> mem 0x80000000-0x800fffff,0x80100000-0x801fffff on ofwbus0
gic0: pn 0x2, arch 0x2, rev 0x1, implementer 0x43b irqs 258
gicv2m0: <ARM Generic Interrupt Controller MSI/MSI-X> mem 0x80200000-0x8020ffff on gic0
gpio0: <ARM PL061 GPIO Controller> mem 0x90300000-0x9030ffff irq 32 on ofwbus0
gpiobus0: <GPIO bus> on gpio0
generic_timer0: <ARMv8 Generic Timer> irq 36,37,38,39 on ofwbus0
Timecounter "ARM MPCore Timecounter" frequency 50000000 Hz quality 1000
Event timer "ARM MPCore Eventtimer" frequency 50000000 Hz quality 1000
gpio0: <GPIO controller> on gpio0
pci0: <Generic PCI host controller> mem 0x4010000000-0x401ffffff on ofwbus0
pci0: <OFW PCI bus> on pci0
virtio_pci0: <VirtIO PCI (legacy) Network adapter> irq 40 at device 1.0 on pci0
vtnet0: <VirtIO Networking Adapter> on virtio_pci0
vtnet0: Ethernet address: 52:54:00:12:34:56
virtio_pci1: <VirtIO PCI (modern) SCMI adapter> irq 41 at device 2.0 on pci0
vtscmi0: <VirtIO SCMI Adapter> on virtio_pci1
vtscmi0: === HAS_P2A:1 HAS_SHARED:0
virtio_pci2: <VirtIO PCI (legacy) Block adapter> irq 42 at device 3.0 on pci0
vtblk0: <VirtIO Block Adapter> on virtio_pci2
vtblk0: 6176MB (12649501 512 byte sectors)
pl0310: <PL031 RTC> mem 0x90100000-0x9010ffff irq 33 on ofwbus0
pl0310: registered as a time-of-day clock, resolution 1.000000s
uart0: <PrimeCell UART (PL011)> mem 0x90000000-0x9000ffff irq 34 on ofwbus0
uart0: console (115200,n,8,1)
pmu0: <Performance Monitoring Units> irq 35 on ofwbus0
cpulist0: <Open Firmware CPU Group> on ofwbus0
cpu0: <Open Firmware CPU> on cpulist0
scmi_virtio0: <ARM SCMI VirtIO Transport driver> on ofw_firmware0
scmi_virtio0: Fed 1 initial P2A buffers to platform.
vtscmi0: Enabled interrupts on VQ[0].
scmi_virtio0: VirtIO cmdq virtqueue configured - cmdq_sz:2
vtscmi0: Enabled interrupts on VQ[1].
scmi_virtio0: VirtIO evtq virtqueue configured - evtq_sz:1
scmi_virtio0: Transport reply timeout initialized to 500ms
scmi_clk0: <SCMI Clock Management Unit> on scmi_virtio0
scmi_virtio0: Sending HDR [5001] token: 0x0 polling:1
scmi_virtio0: Dropping inflight request for token: 0x0.
scmi_clk0: Found 4 clocks.
scmi_virtio0: Sending HDR [45003] token: 0x1 polling:1
scmi_virtio0: Dropping inflight request for token: 0x1.
scmi_clk0: Clock 'VPU' registered.
scmi_virtio0: Sending HDR [85003] token: 0x2 polling:1
scmi_virtio0: Dropping inflight request for token: 0x2.
scmi_clk0: Clock 'DPU' registered.
scmi_virtio0: Sending HDR [5003] token: 0x3 polling:1
scmi_virtio0: Dropping inflight request for token: 0x3.
scmi_clk0: Clock 'PIXEL_0' registered.
scmi_virtio0: Sending HDR [108003] token: 0x4 polling:1
scmi_virtio0: Dropping inflight request for token: 0x4.
scmi_clk0: Clock 'PIXEL_1' registered.
armv8crypto0: <AES-CBC,AES-XTS,AES-GCM>
Timecounters tick every 1.000 msec
usb_needs_explore_all: no devclass
CPU 0: ARM Cortex-A57 r0p0 affinity: 0
    Cache Type = <64 byte D-cacheline,64 byte I-cacheline,VIPT ICache,64 byte ERG,64 byte CWG>
    Instruction Set Attributes 0 = <CRC32,SHA2,SHA1,AES+PMULL>
    Instruction Set Attributes 1 = <>
    Trying to mount root from ufs:ufs/rootfs []...
    Instruction Set Attributes 2 = <>
```

```
root@deb-buster-arm64:~# ./SCP_FW.elf
[ 0.000000]
[ 0.000000]
[ 0.000000]
[ 0.000000]
[ 0.000000]
[ 0.000000] v2.8.0-2021-05-21_18-43-59-v2.4.0-1507-g72e92fdb-dirty
[ 0.000000]
[ 0.000000] [FWK] Module initialization complete!
** Message: 11:56:43.897: entering main loop, awaiting messages
** Message: 11:56:43.903: awaiting connection to vscmi-ospml.sock
** Message: 11:56:43.903: awaiting connection to vscmi-ospm0.sock
===== Vhost user message =====
Request: VHOST_USER_GET_FEATURES (1)
Flags: 0x1
Size: 0
Sending back to guest u64: 0x0000000175000001
===== Vhost user message =====
Request: VHOST_USER_GET_PROTOCOL_FEATURES (15)
Flags: 0x1
Size: 0
===== Vhost user message =====
Request: VHOST_USER_SET_PROTOCOL_FEATURES (16)
Flags: 0x1
Size: 8
u64: 0x00000000000000c2b
===== Vhost user message =====
Request: VHOST_USER_GET_QUEUE_NUM (17)
Flags: 0x1
Size: 0
===== Vhost user message =====
Request: VHOST_USER_GET_MAX_MEM_SLOTS (36)
Flags: 0x1
Size: 0
u64: 0x0000000000000020
===== Vhost user message =====
Request: VHOST_USER_SET_SLAVE_REQ_FD (21)
Flags: 0x9
Size: 0
Fds: 7
Got slave_fd: 7
===== Vhost user message =====
Request: VHOST_USER_SET_OWNER (3)
Flags: 0x1
Size: 0
===== Vhost user message =====
Request: VHOST_USER_GET_FEATURES (1)
Flags: 0x1
Size: 0
Sending back to guest u64: 0x0000000175000001
===== Vhost user message =====
Request: VHOST_USER_SET_VRING_CALL (13)
Flags: 0x1
Size: 8
Fds: 8
u64: 0x0000000000000000
Got call_fd: 8 for vq: 0
===== Vhost user message =====
Request: VHOST_USER_SET_VRING_CALL (13)
Flags: 0x1
Size: 8
Fds: 9
u64: 0x0000000000000001
Got call_fd: 9 for vq: 1
===== Vhost user message =====
Request: VHOST_USER_SET_VRING_CALL (13)
Flags: 0x1
Size: 8
Fds: 10
u64: 0x0000000000000000
Got call_fd: 10 for vq: 0
===== Vhost user message =====
Request: VHOST_USER_SET_VRING_CALL (13)
Flags: 0x1
Size: 8
Fds: 8
u64: 0x0000000000000001
Got call_fd: 8 for vq: 1
===== Vhost user message =====
Request: VHOST_USER_SET_FEATURES (2)
Flags: 0x1
```

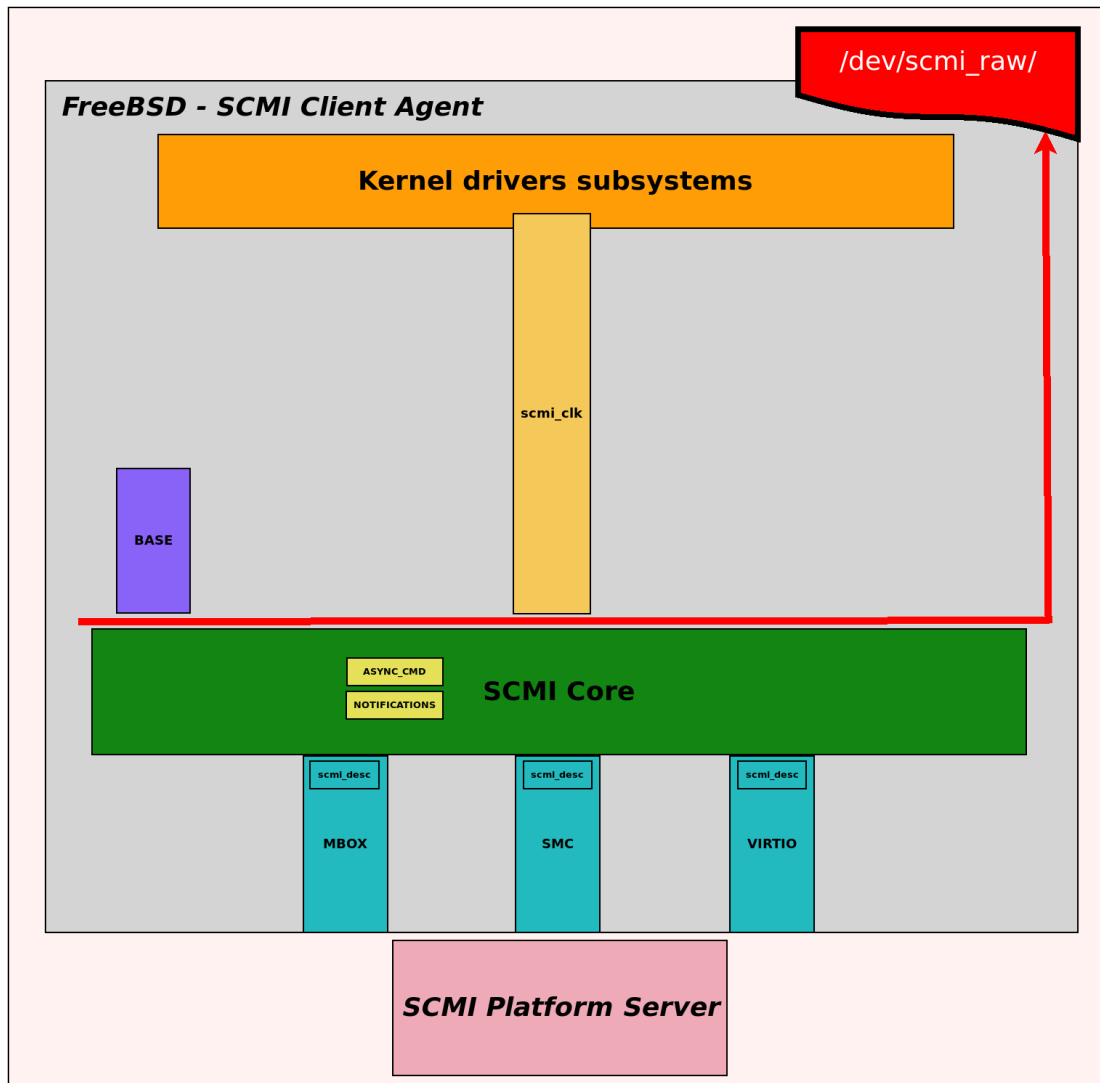
FreeBSD SCMI Stack - III



FreeBSD SCMI Support – WIP or planned

- Adding **SCMI RAW** for message injection to enable:
 - **testing** of the SCMI Core stack
 - **compliance** testing of a real Platform server
- Adding SCMI Core support for:
 - **Asynchronous commands**
 - **Notifications**
- Adding **Base** protocol
- Some **SDT(?)** traces to dump ongoing SCMI transactions

FreeBSD SCMI Stack - IV



FreeBSD SCMI Support - NEXT

The current WIP will enable a more complete development and test environment for SCMI on FreeBSD...so...

What's NEXT ?

- Per-protocol transport channels (where possible)
- Protocol layer abstraction (maybe)
- Vendor Protocols support
- SCMI Test Driver
- More transports (if really required by new HW)

Enabling more SCMI Protocols and related drivers ?

Depends on the availability of related Kernel frameworks on FreeBSD:

- Power
- SysPower
- Performance: cpufreq(4) / scmi_perf ?
- Sensors ?
- Reset ?
- Voltage ?
- Powercap ?
- Pincontrol ?

Questions ?

arm

Thank You

Danke

Gracias

Grazie

谢谢

ありがとう

Asante

Merci

감사합니다

धन्यवाद

Kiitos

شكرًا

धन्यवाद

תודה

ధన్యవాదములు



The Arm trademarks featured in this presentation are registered trademarks or trademarks of Arm Limited (or its subsidiaries) in the US and/or elsewhere. All rights reserved. All other marks featured may be trademarks of their respective owners.

www.arm.com/company/policies/trademarks