arm

TF-M Dual-cpu NS Mailbox Improvement

Enhance integration with NS environment

David Hu
2020 Nov

Agenda

- TF-M dual-cpu NS mailbox enhancement
  - Simplify NS RTOS port
- Enhance NS mailbox working model
  - A new working model to support NS applications isolation
  - Refine working model configuration

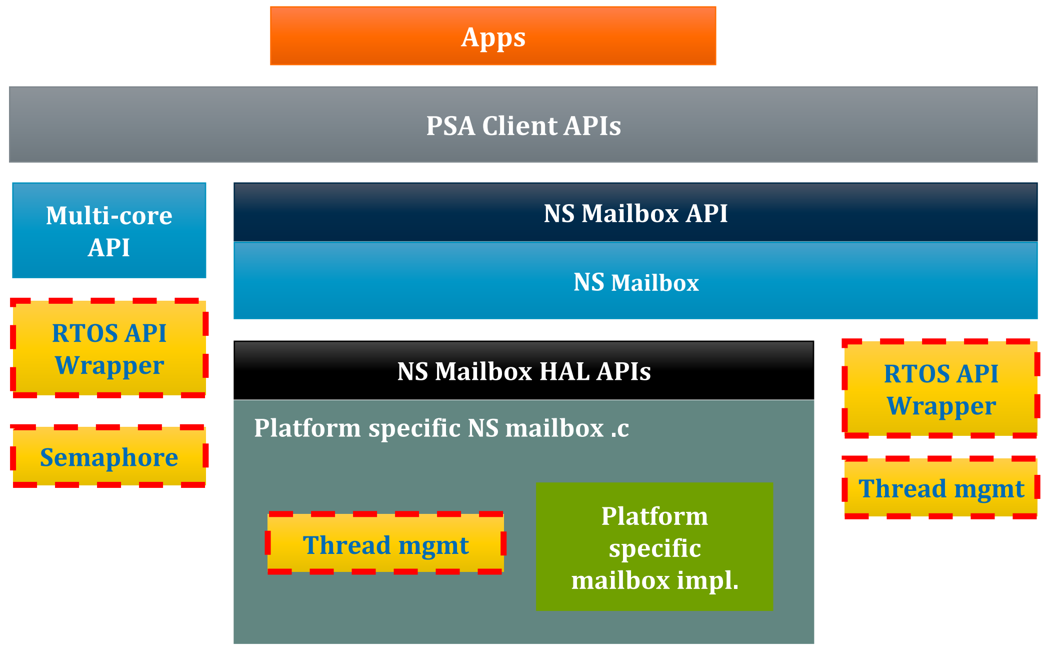
NS mailbox enhancement

Various RTOS modules distributed in TF-M

- Semaphores inside PSA Client API implementation
  - A dedicated API set outside NS mailbox

```
uint32_t psa_framework_version(void)
{
    if (tfm_ns_multi_core_lock_acquire() != OS_WRAPPER_SUCCESS) {
        return PSA_VERSION_NONE;
    }

    /* mailbox handling */

    if (tfm_ns_multi_core_lock_release() != OS_WRAPPER_SUCCESS) {
        return PSA_VERSION_NONE;
    }
}
```

- Part of thread mgmt. placed in platform specific driver
  - RTOS specific thread mgmt.
  - Platform independent
- Improve goals:
  - Sort out mailbox interface
  - Improve dependencies on RTOS



Simplify NS mailbox API

- A single NS mailbox API `tfm_ns_mailbox_client_call()`
  - Combine various NS mailbox APIs
  - Avoid exporting NS mailbox internal variables

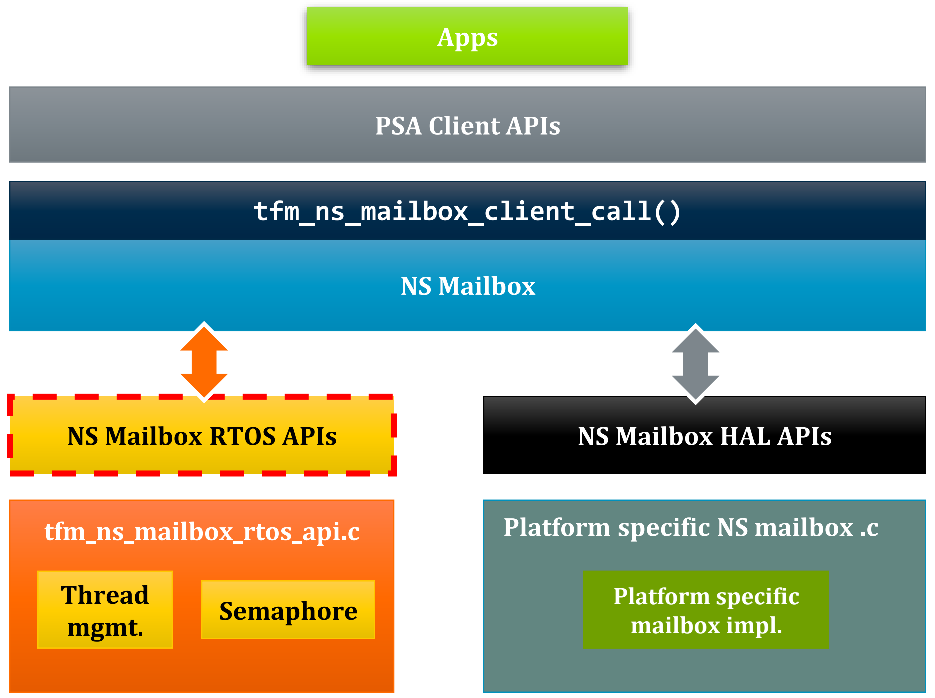
```
uint32_t psa_framework_version(void)
{
    mailbox_msg_handle_t handle;
    ...

    if (tfm_ns_multi_core_lock_acquire() != OS_WRAPPER_SUCCESS) {
        return PSA_VERSION_NONE;
    }

    handle = tfm_ns_mailbox_tx_client_req(...);
    ...

    mailbox_wait_reply(handle);

    ret = tfm_ns_mailbox_rx_client_reply(handle, ...);
    ...

    if (tfm_ns_multi_core_lock_release() != OS_WRAPPER_SUCCESS) {
        return PSA_VERSION_NONE;
    }

    return version;
}
```

```
uint32_t psa_framework_version(void)
{
    ...

    ret = tfm_ns_mailbox_client_call(...);
    ...

    return version;
}
```

Re-organize NS mailbox dependencies on RTOS

Decouple ROTS specific impl. from Platform and common NS mailbox

- Define NS mailbox RTOS API
  - `tfm_ns_mailbox_os_xxx()`
  - Decoupled from platform HAL and common NS mailbox
    - Thread mgmt. is moved out from platform impl.
    - Semaphores are moved out from multi-core API
- NS mailbox RTOS APIs implementation
  - `tfm_ns_mailbox_rtos_api.c` as a reference
    - Can be directly replaced with RTOS specific impl.
  - RTOS API wrapper becomes optional

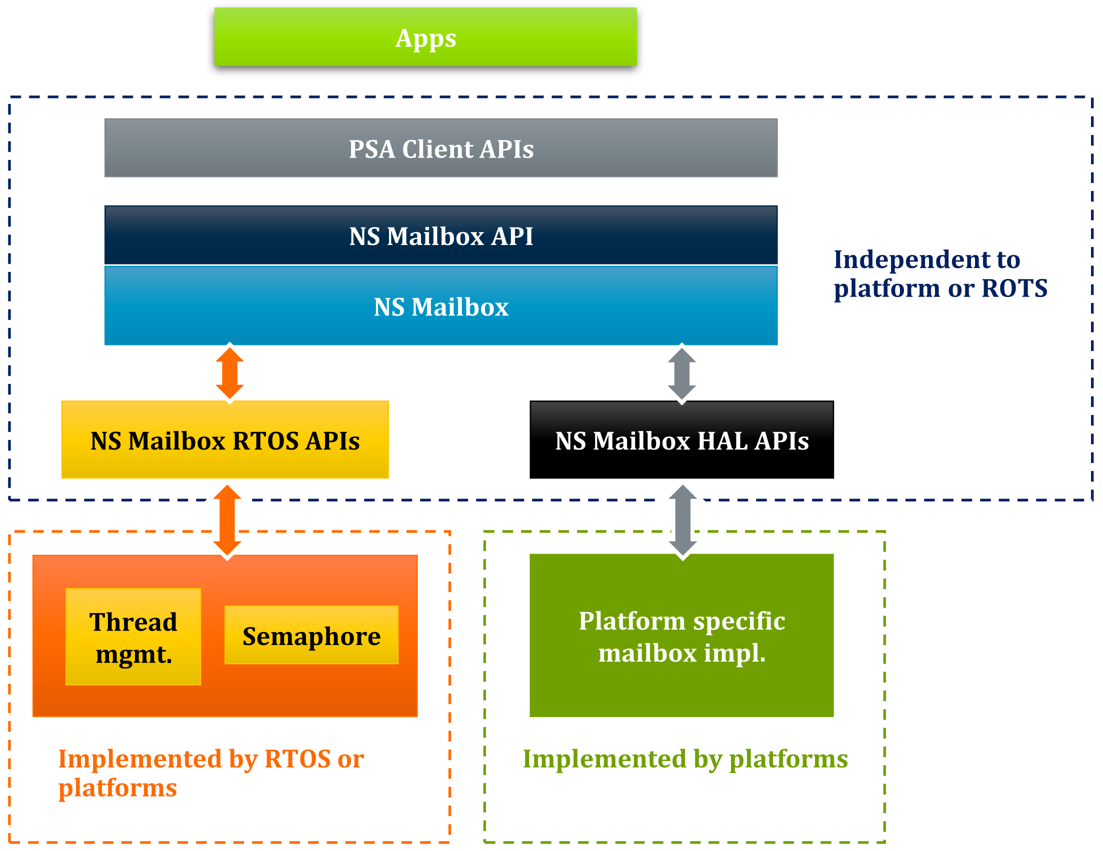


Apps
PSA Client APIs
tfm_ns_mailbox_client_call()
NS Mailbox
NS Mailbox RTOS APIs
NS Mailbox HAL APIs
tfm_ns_mailbox_rtos_api.c
Thread mgmt.
Semaphore
Platform specific NS mailbox .c
Platform specific mailbox impl.

Easier integration with RTOS on platforms

- Platform specific mailbox HW impl.
  - Implemented by platform partners
  - Implemented under platform folder in TF-M

- RTOS support to NS mailbox
  - Implemented by application developers or platform partners according to actual usage scenarios
  - Maintain a dedicated .c for each RTOS

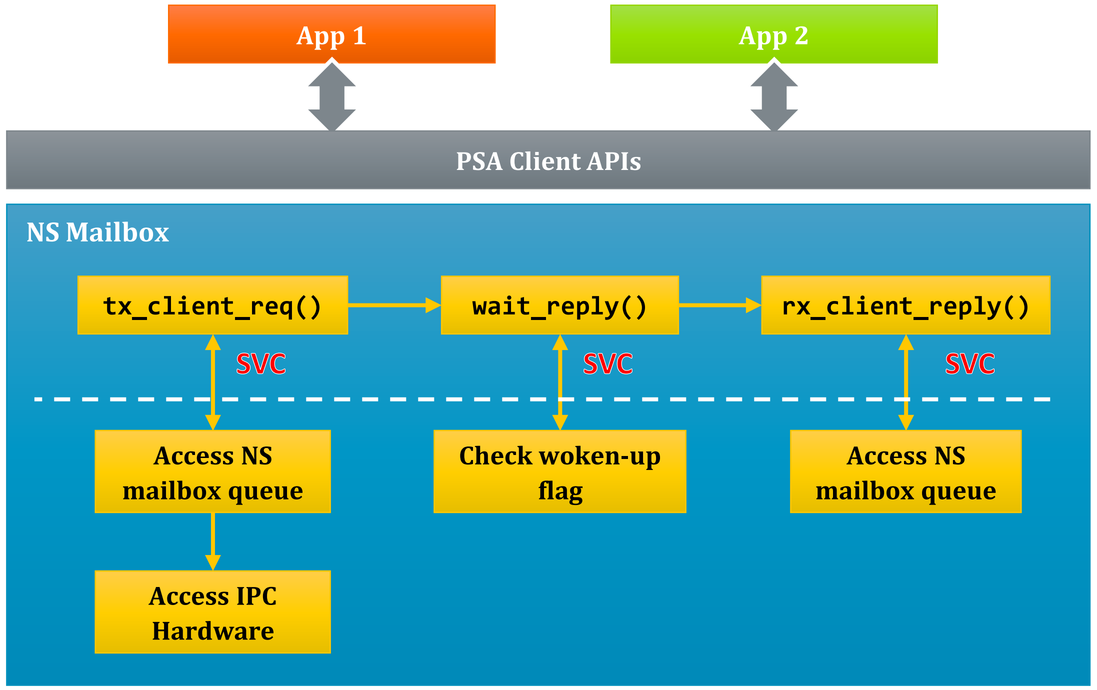


arm

Enhance NS mailbox working model

Extra port effort in a more complex usage scenario

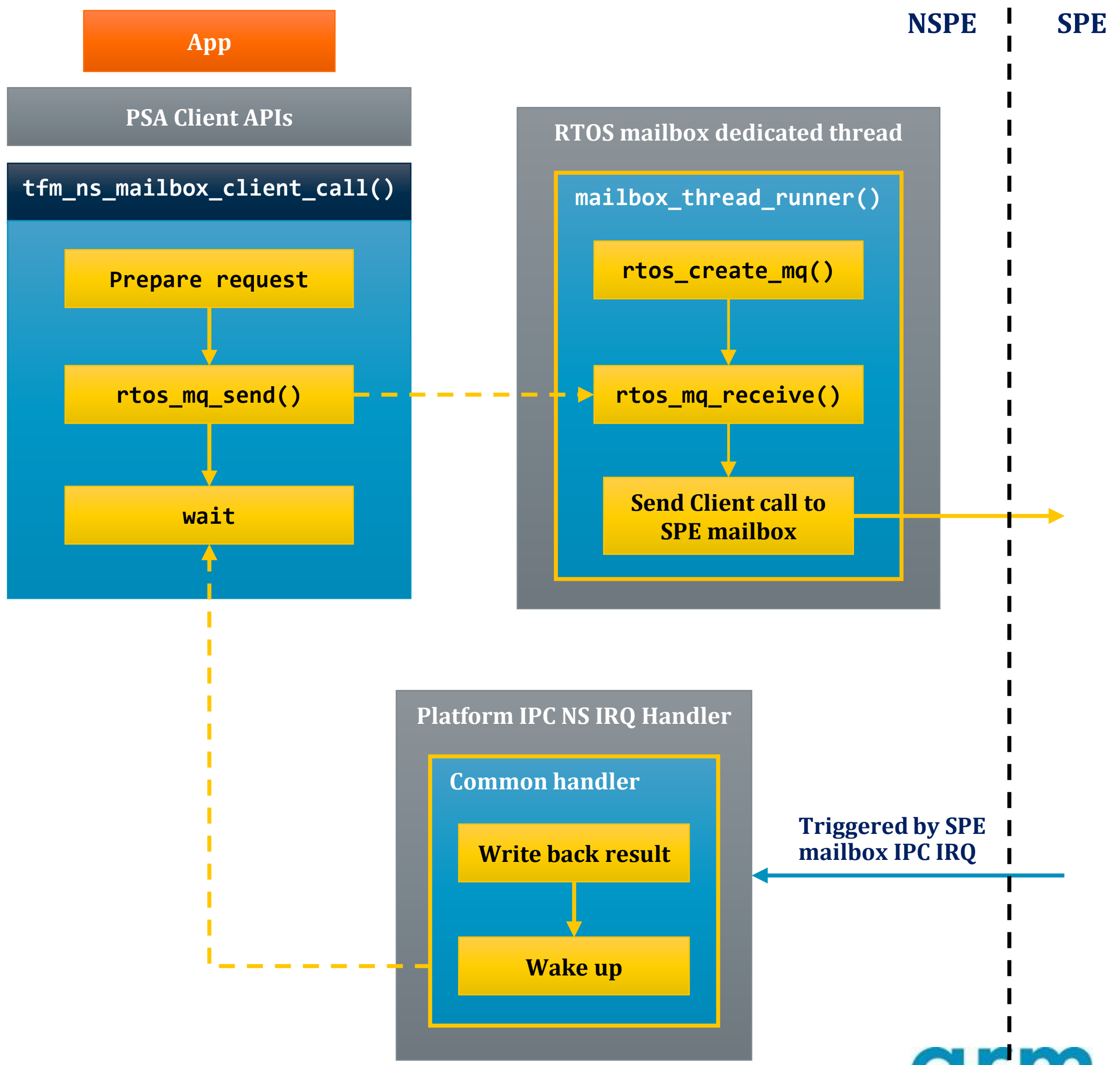
- NS MPU enabled to isolate NS applications
  - Difficult to specify mailbox static objects addresses in application thread MPU regions
  - Modify common NS mailbox impl. to insert SVCs
  - Different SVC handler hacks in various RTOSs
- Goal: Support NS thread isolation more easily



A new NS mailbox working model

Besides existing NS mailbox implementation

- A dedicated NS mailbox thread assigned
  - Execute `mailbox_thread_runner()`
  - Receive requests from application threads via RTOS message queue
  - Wait if mailbox queue is full
- Simplify RTOS port for thread isolation
  - No explicit SVC is required
    - No longer necessary to hack common NS mailbox or RTOS
  - Get rid of semaphores
- Fit more in RTOS/OS thread mgmt.
  - Mailbox dedicated thread can run in privileged mode
  - Application threads can be isolated
    - Each thread maintains its own set of resource during NS mailbox handling



Refine NS mailbox working model configuration

- Clarify the responsibilities of platform implementation and NS integration

TFM_MULTI_CORE_NS_OS
TFM_MULTI_CORE_OS_MAILBOX_THREAD
- Controlled by NS integration
- NS environment

NUM_MAILBOX_QUEUE_SLOT
- Defined by platform and SPE
- Hardware resource

	TFM_MULTI_CORE_NS_OS *OFF*	TFM_MULTI_CORE_NS_OS *ON* TFM_MULTI_CORE_OS_MAILBOX_THREAD *OFF*	TFM_MULTI_CORE_NS_OS *ON* TFM_MULTI_CORE_OS_MAILBOX_THREAD *ON*
NUM_MAILBOX_QUEUE_SLOT > 1	----	• NS OS environment • *Enable* multiple PSA NS client call feature	• NS OS environment • Dedicated NS mailbox thread • *Enable* multiple PSA NS client call feature
NUM_MAILBOX_QUEUE_SLOT == 1	• NS bare metal environment	• NS OS environment	• NS OS environment • Dedicated NS mailbox thread

TFM_MULTI_CORE_MULTI_CLIENT_CALL
- Controlled by SPE/NSPE

NUM_MAILBOX_QUEUE_SLOT
- Defined by platform and SPE

	TFM_MULTI_CORE_MULTI_CLIENT_CALL *ON*	TFM_MULTI_CORE_MULTI_CLIENT_CALL *OFF*
NUM_MAILBOX_QUEUE_SLOT > 1	• Enable multiple PSA NS client call feature • Rely on platform IPC interrupt	----
NUM_MAILBOX_QUEUE_SLOT == 1	----	• Disable multiple PSA NS client call feature • Looping mailbox flag

Current status

Current status

- Patches under review
  - Enhancement
  - New NS mailbox working model
- Collecting feedback from partners for actual usage scenarios
  - Comments and suggestions are welcome
- NS mailbox enhancement
  - Looking forward to achieving approval
- NS mailbox working model with a dedicated thread
  - Further discussion if necessary

arm

Thank You
Danke
Merci
谢谢
ありがとう
Gracias
Kiitos
감사합니다
धन्यवाद
شكرًا
ধন্যবাদ
תודה

Backup slides

Quantitative results comparison

- Latency or throughput is not affected in this proposal
  - Compared to current implementation
  - *Although performance is not the main purpose in this proposal*

		Current Impl.	***Enhancement***	***Dedicated thread***
Lightweight test	Total nr of threads	7	7	7
	Nr of pending slots in average	2.5	2.9	1.7
	Ticks cost in each PSA client call	0.2	0.1	8.5
	Ticks cost in total	820	715	30761
Heavyweight test	Total nr of threads	5	5	5
	Nr of pending slots in average	3.8	3.9	3.8
	Ticks cost in each round	699.3	697.1	698.1
	Ticks cost in total	335710	334632	335109
Out-of-Order test	Total nr of threads	5	5	5
	Nr of pending slots in average	2.8	2.8	2.6
	Ticks cost in each round	11.8	11.0	12.0
	Ticks cost in total	31450	29452	32000

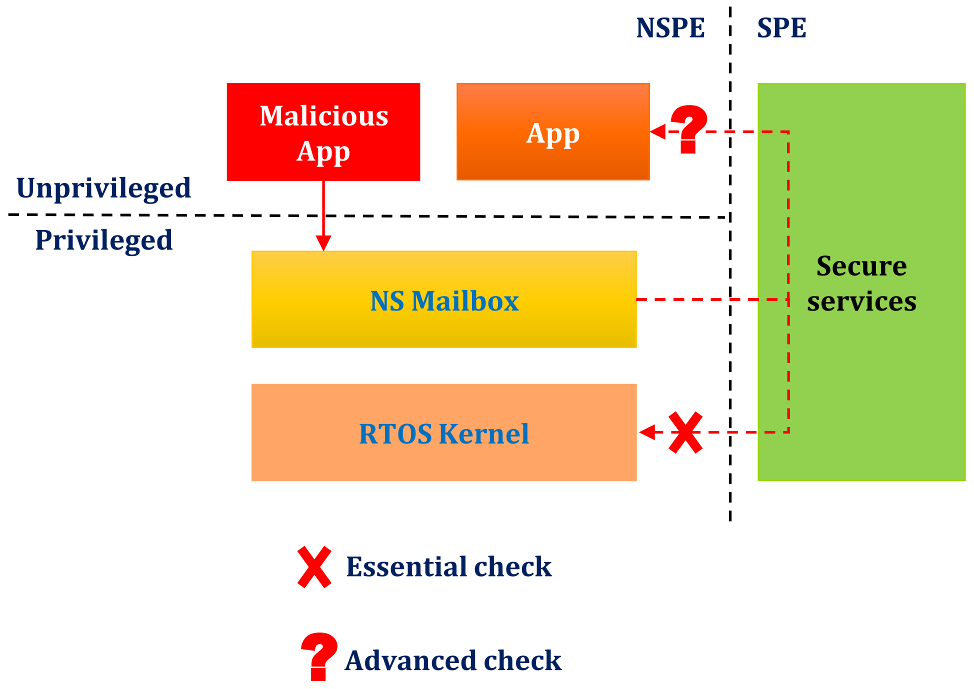
Based on TF-Mv1.2.0
TF-M multi-core tests running on Cypress PSoC 64
Total 4 mailbox queue slots

Further security consideration

Not implemented yet

- "Boomerang" attack
  - SPE is unaware of corresponding NSPE isolation configs
  - NS malicious app cheats SPE to access other NS thread area or NS privileged area, bypassing NSPE MPU HW

- NS memory check in NS mailbox
  - *If required by usage scenario thread model*
  - Essential check: an unprivileged app provides addresses belonging to privileged areas.
  - Advanced check: an app provides addresses not belonging to itself
    - Highly depends on platform and RTOS impl.


NSPE
SPE
Malicious App
App
Unprivileged
Privileged
NS Mailbox
RTOS Kernel
Secure services
Essential check
Advanced check