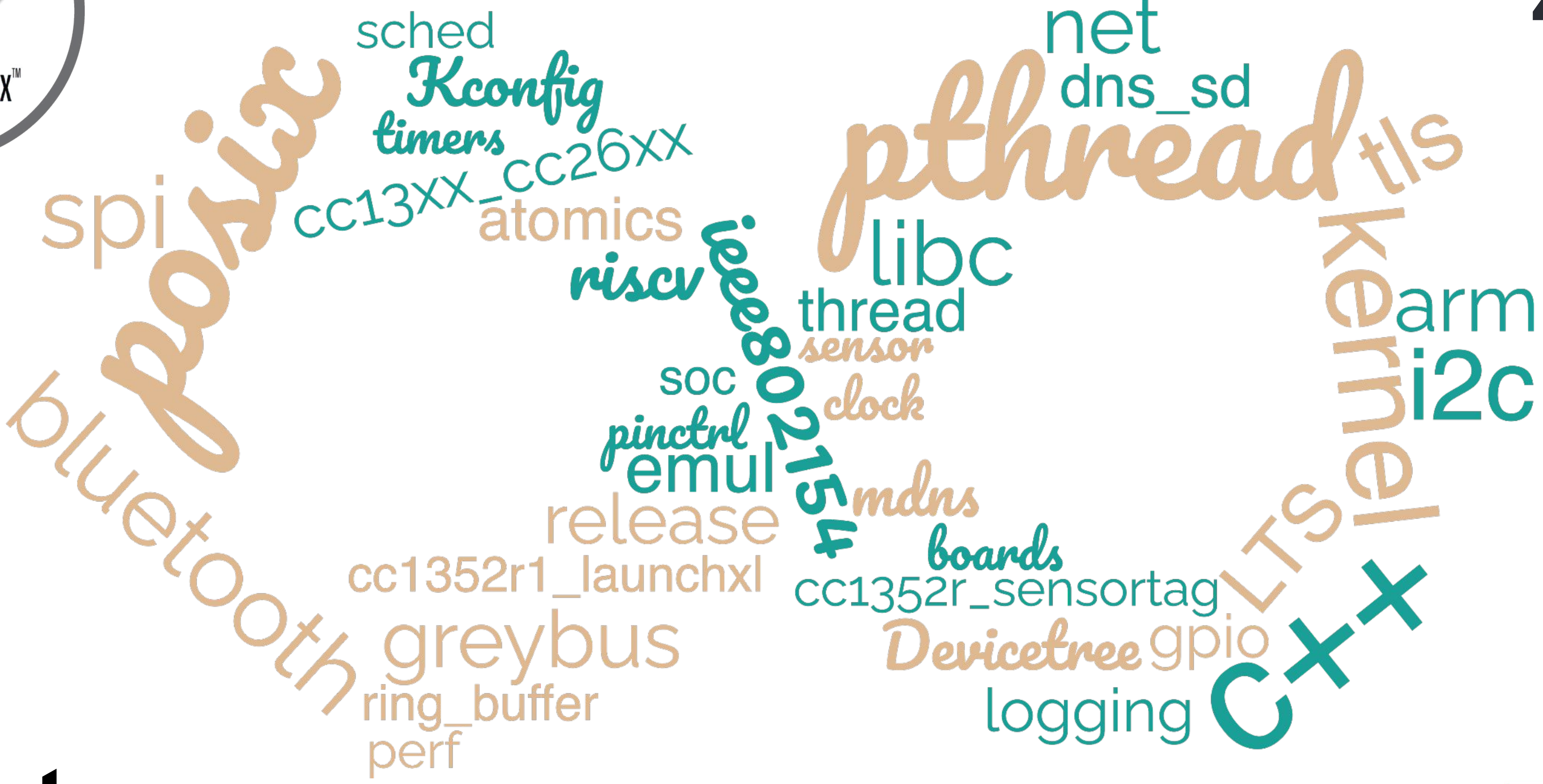
POSIX Roadmap for Zephyr LTSv3

2023-06-29: Embedded Open Source Summit

Chris Friedt
Embedded SWE, Meta
Zephyr POSIX API Maintainer

gentoo linux™

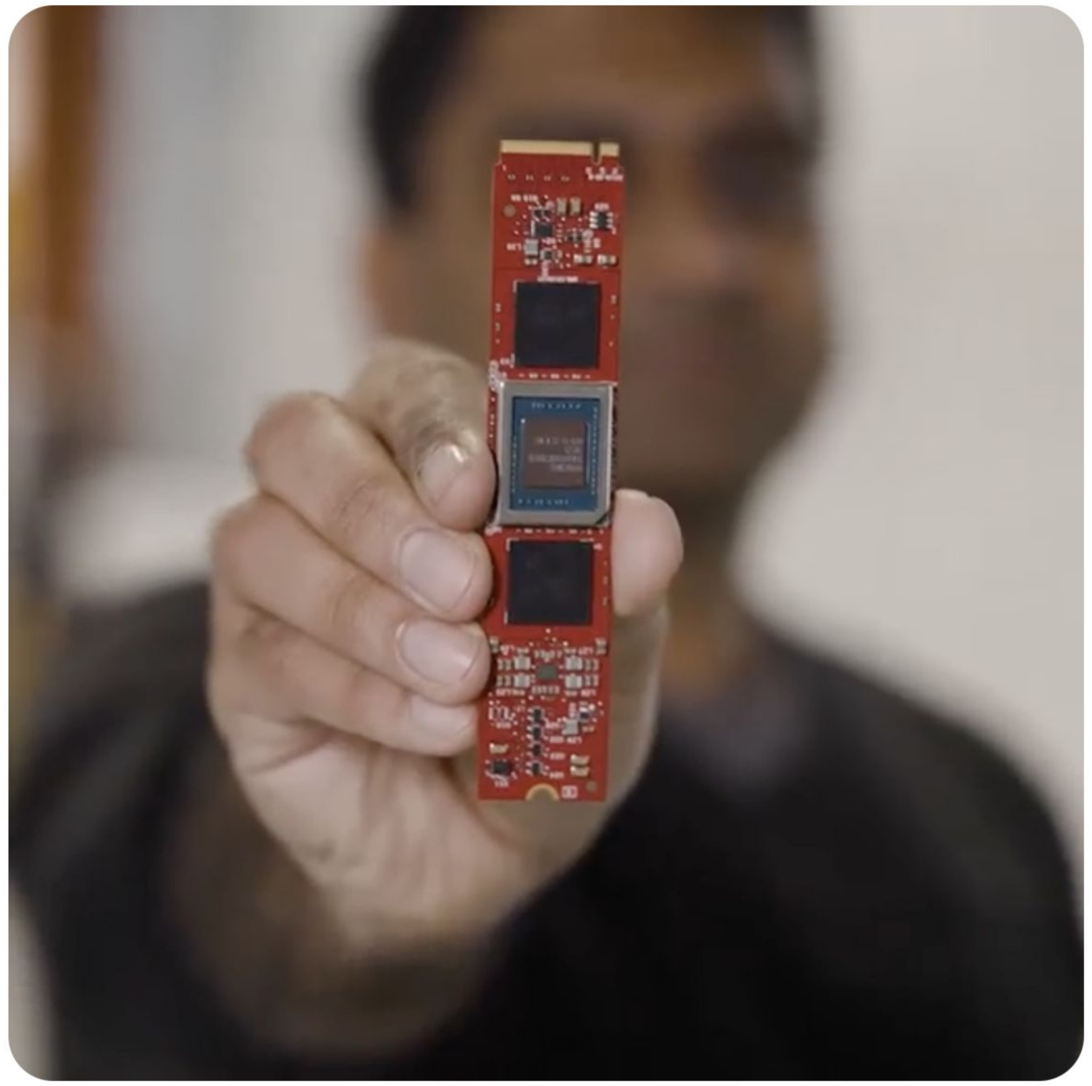
Zephyr®



yocto PROJECT

How does Meta∞ use Zephyr🪁?

- Meta Scalable Video Processor (MSVP)
- Why video transcode ASICs?
- 4B videos / day on Facebook
- Power, Storage, Performance
- 9x faster throughput for H264
- 50x faster throughput for VP9
- 6x better performance for HQ VOD
- 50% less power consumption
- AV1 Coming Soon..



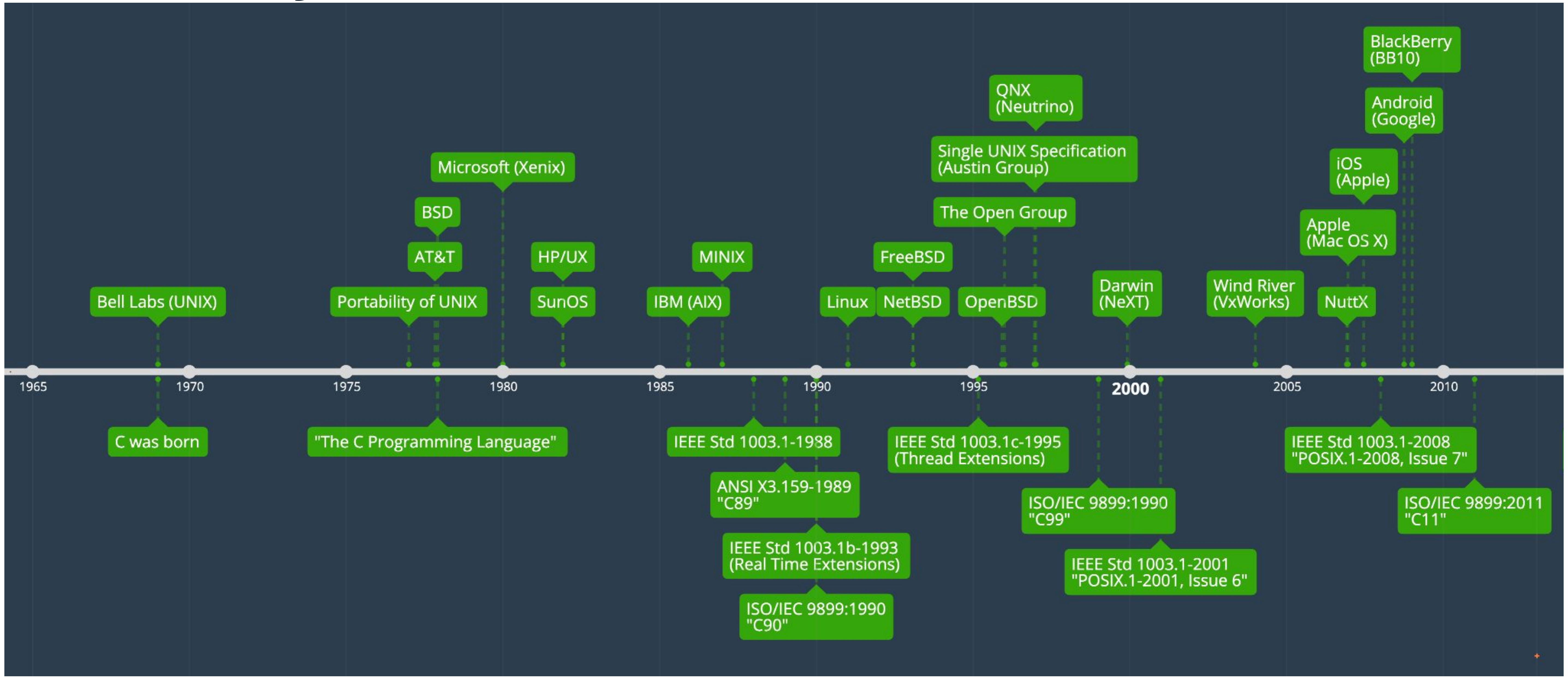
How does Meta∞ use Zephyr🪁?

- Meta Training and Inference Accelerator (MTIA)
- Why AI ASICs?
- Feeds, Ads, Content, Ranking..
- DLRM Models: 4.5 GB up to 750 GB
- Power, Storage, Performance
- 2x Efficiency of today's GPUs
- PyTorch 2.0
- MTIA @ ISCA 2023

Agenda

01 Overview of POSIX in Zephyr

The Way Back Machine..

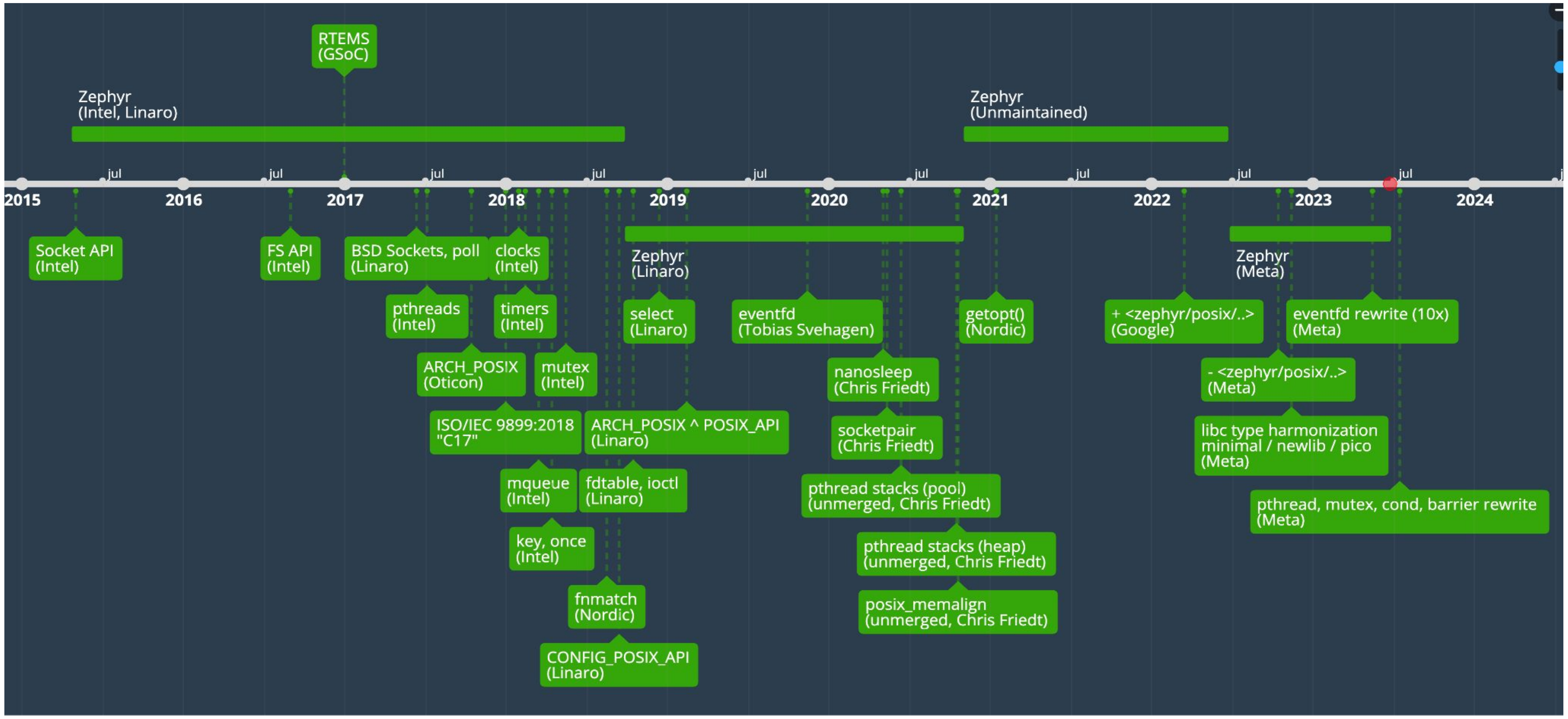


POSIX Turns 35 Years Old!

Bell Labs UNIX Turns 50 Years Old!*

* 50 years since UNIX was announced outside of Bell Labs

History of POSIX in Zephyr


RTEMS (GSoC)
Zephyr (Intel, Linaro)
Zephyr (Unmaintained)
2015
2016
2017
2018
2019
2020
2021
2022
2023
2024
jul
Socket API (Intel)
FS API (Intel)
BSD Sockets, poll (Linaro)
clocks (Intel)
pthreads (Intel)
timers (Intel)
ARCH_POSIX (Oticon)
mutex (Intel)
ISO/IEC 9899:2018 "C17"
ARCH_POSIX ^ POSIX_API (Linaro)
mqueue (Intel)
fdtable, ioctl (Linaro)
key, once (Intel)
fnmatch (Nordic)
CONFIG_POSIX_API (Linaro)
Zephyr (Linaro)
select (Linaro)
eventfd (Tobias Svehagen)
nanosleep (Chris Friedt)
socketpair (Chris Friedt)
pthread stacks (pool) (unmerged, Chris Friedt)
pthread stacks (heap) (unmerged, Chris Friedt)
posix_memalign (unmerged, Chris Friedt)
getopt() (Nordic)
+ <zephyr/posix/..> (Google)
Zephyr (Meta)
eventfd rewrite (10x) (Meta)
- <zephyr/posix/..> (Meta)
libc type harmonization minimal / newlib / pico (Meta)
pthread, mutex, cond, barrier rewrite (Meta)

Why POSIX?

- Portability
- Mature API
- Powers 1B 🚗 ¯_(ツ)_/¯
- Powers 2B 💻 🖥️ 🖨️
- Powers 16B Mobile 📱

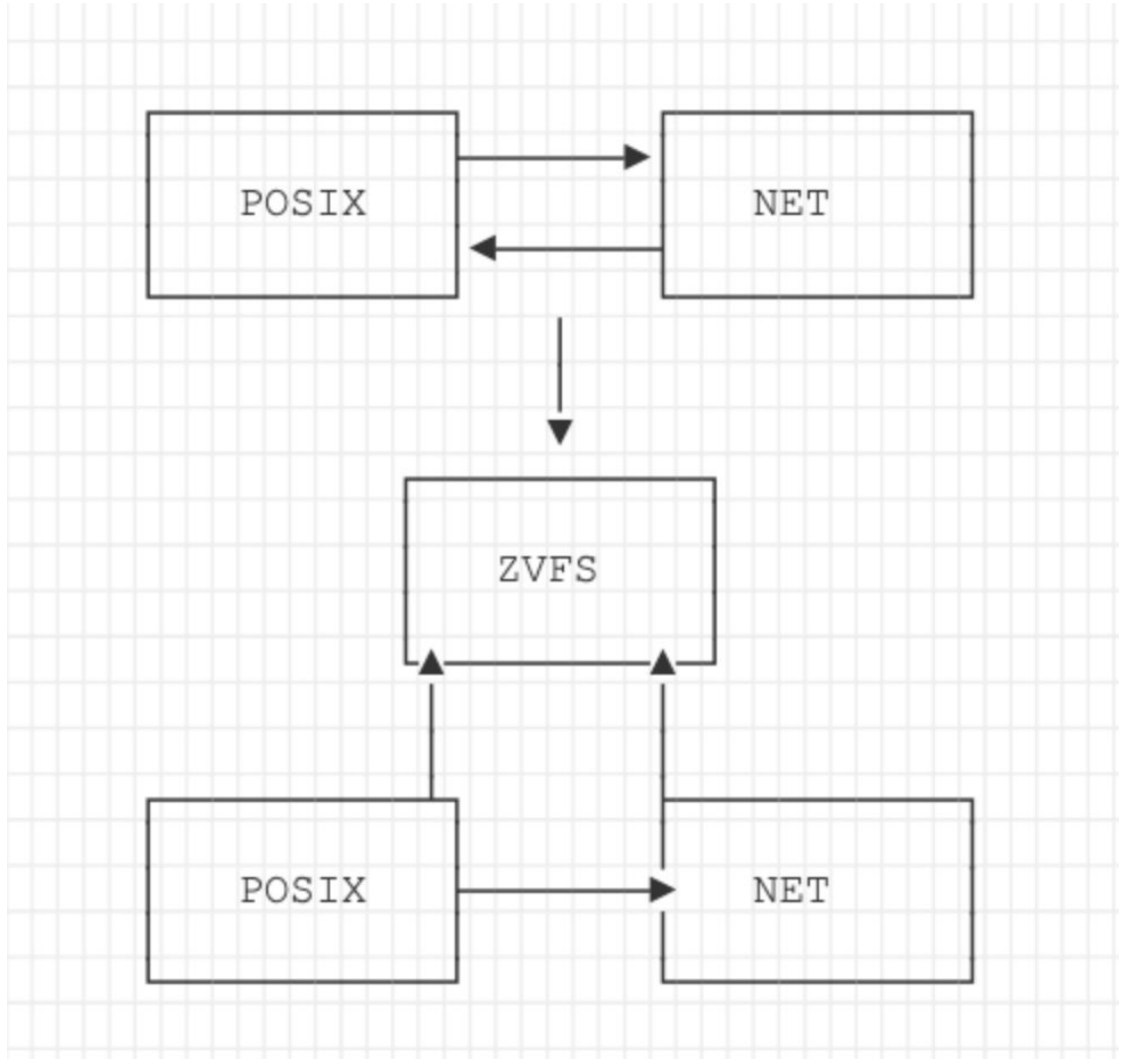
02 Goals for LTSv3

High-Level Goals

1. Improve **Maintainability**
2. Improve the application / libc / toolchain **Interface**
3. Improve application / library **Portability**

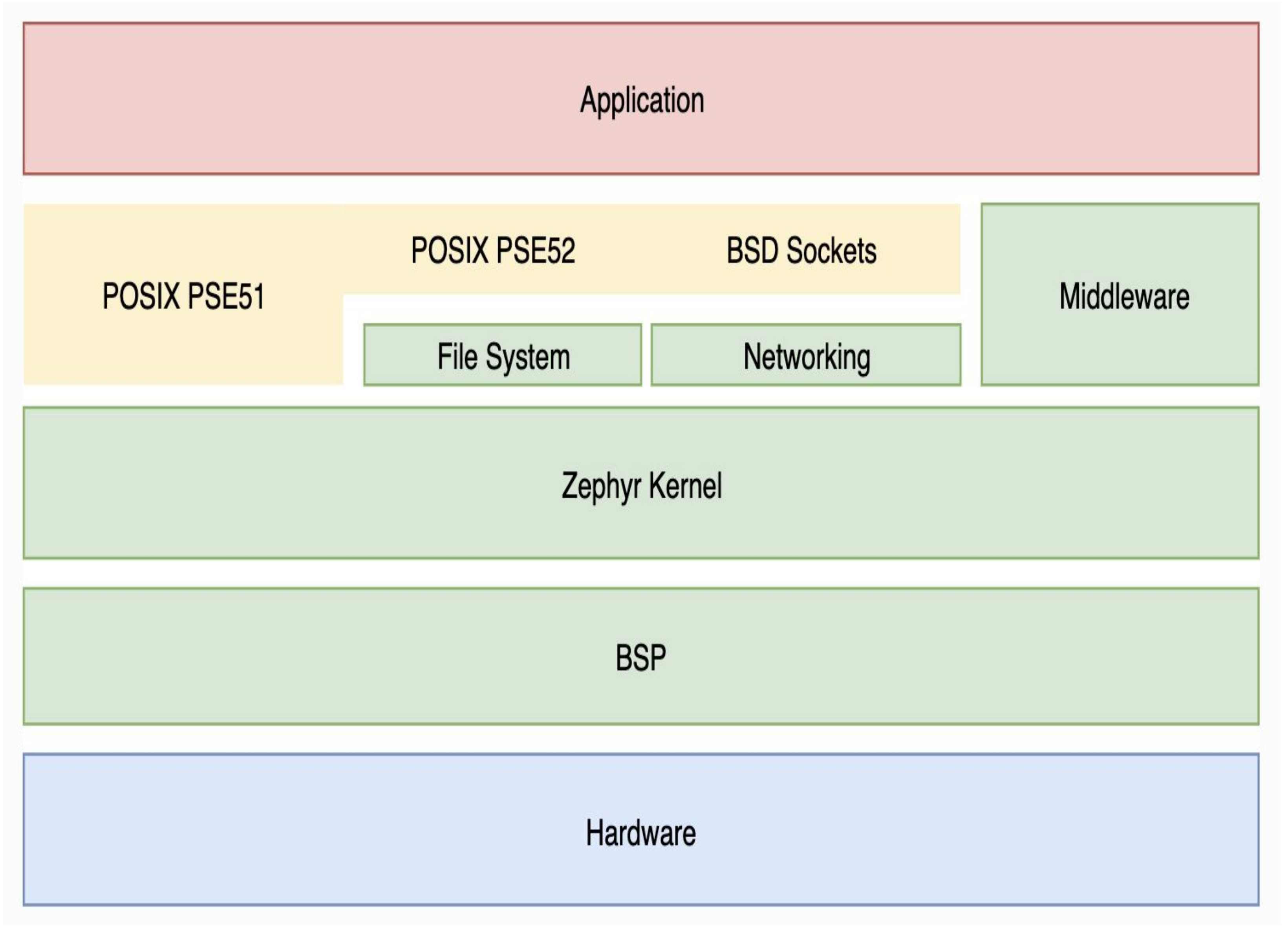
Maintainability

- Abstract POSIX resources as integers (common representation among libcs)
  - e.g. *pthread_mutex_t*, *pthread_t*
- Re-use Zephyr synchronization primitives within POSIX
  - E.g. use *k_mutex* internally rather than dogfooding *pthread_mutex_t*
- fdtable -> zvfs
  - Common library that can be referenced by *subsys/net*, POSIX, etc
- ARCH_POSIX ^ POSIX_API
  - ARCH_POSIX Maintainer -> RFC #58305 🙏



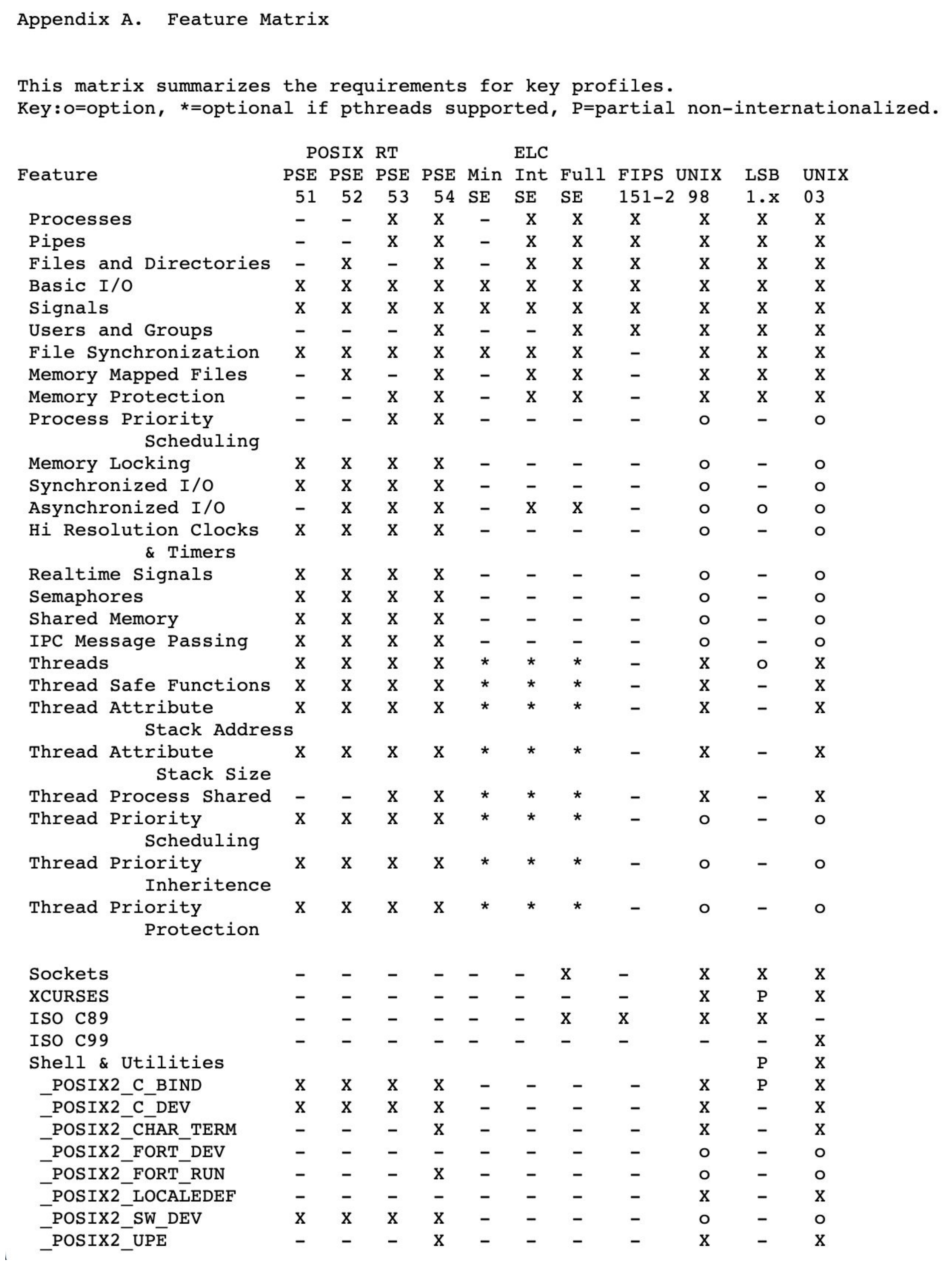
Interface

- **Important**: POSIX is an *interface* not a *subsystem*
  - Any library code required to support POSIX should be
    - Part of Zephyr itself
    - As minimal as necessary to support the *interface*
  - Must remember to avoid layering violations
- Support standard include paths for 3rd-party applications and libraries E.g. *<unistd.h>* rather than *<posix/zephyr/unistd.h>*
- Kconfig for POSIX feature test macros. _POSIX_TIMERS
  - Standard interface for libc / toolchain headers
- Support POSIX with external libc / toolchain (e.g. IAR)
  - Zephyr *must* supply POSIX declarations when the libc does not



Portability: PSE51

- Ref: IEEE 1003.1-2017
- PSE51: Minimal Real-time System Profile
  - Single, multi-threaded process, no file system, no user or group support, selected options
  - Detailed in IEEE Std 1003.13, also see man 7 posixoptions
- Compilation Environment:
  - *<unistd.h>*
    - *#define _POSIX_AEP_REALTIME_MINIMAL 200312L*
    - *#define _POSIX_AEP_REALTIME_LANG_C99*
- Options Requirements:
  - _POSIX_C_LANG_JUMP, _POSIX_SIGNALS, _POSIX_SINGLE_PROCESS, _POSIX_THREADS_BASE, _POSIX_CLOCK_SELECTION, _POSIX_FSYNC, _POSIX_MEMLOCK, _POSIX_MEMLOCK_RANGE, _POSIX_REALTIME_SIGNALS, _POSIX_SHARED_MEMORY_OBJECTS, _POSIX_SYNCHRONIZED_IO, _POSIX_CPUTIME, _POSIX_THREAD_PRIO_INHERIT, _POSIX_THREAD_PRIO_PROTECT, _POSIX_TIMEOUTS, _POSIX_TIMERS
- Exceptions:
  - POSIX_DEVICE_IO *FILE* ops, *scanf(), vscanf()*
  - POSIX_SINGLE_PROCESS *confstr(), *env()*
  - _POSIX_SPORADIC_SERVER (kernel?)



Appendix A. Feature Matrix

This matrix summarizes the requirements for key profiles.
Key:o=option, *=optional if pthreads supported, P=partial non-internationalized.

	POSIX	RT				ELC					
Feature	PSE 51	PSE 52	PSE 53	PSE 54	Min SE	Int SE	Full SE	FIPS 151-2	UNIX 98	LSB 1.x	UNIX 03
Processes	-	-	X	X	-	X	X	X	X	X	X
Pipes	-	-	X	X	-	X	X	X	X	X	X
Files and Directories	-	X	-	X	-	X	X	X	X	X	X
Basic I/O	X	X	X	X	X	X	X	X	X	X	X
Signals	X	X	X	X	X	X	X	X	X	X	X
Users and Groups	-	-	-	X	-	-	X	X	X	X	X
File Synchronization	X	X	X	X	X	X	X	-	X	X	X
Memory Mapped Files	-	X	-	X	-	X	X	-	X	X	X
Memory Protection	-	-	X	X	-	X	X	-	X	X	X
Process Priority Scheduling	-	-	X	X	-	-	-	-	o	-	o
Memory Locking	X	X	X	X	-	-	-	-	o	-	o
Synchronized I/O	X	X	X	X	-	-	-	-	o	-	o
Asynchronized I/O	-	X	X	X	-	X	X	-	o	o	o
Hi Resolution Clocks & Timers	X	X	X	X	-	-	-	-	o	-	o
Realtime Signals	X	X	X	X	-	-	-	-	o	-	o
Semaphores	X	X	X	X	-	-	-	-	o	-	o
Shared Memory	X	X	X	X	-	-	-	-	o	-	o
IPC Message Passing	X	X	X	X	-	-	-	-	o	-	o
Threads	X	X	X	X	*	*	*	-	X	o	X
Thread Safe Functions	X	X	X	X	*	*	*	-	X	-	X
Thread Attribute Stack Address	X	X	X	X	*	*	*	-	X	-	X
Thread Attribute Stack Size	X	X	X	X	*	*	*	-	X	-	X
Thread Process Shared	-	-	X	X	*	*	*	-	X	-	X
Thread Priority Scheduling	X	X	X	X	*	*	*	-	o	-	o
Thread Priority Inheritence	X	X	X	X	*	*	*	-	o	-	o
Thread Priority Protection	X	X	X	X	*	*	*	-	o	-	o
Sockets	-	-	-	-	-	-	X	-	X	X	X
XCURSES	-	-	-	-	-	-	-	-	X	P	X
ISO C89	-	-	-	-	-	-	X	X	X	X	-
ISO C99	-	-	-	-	-	-	-	-	-	-	X
Shell & Utilities										P	X
_POSIX2_C_BIND	X	X	X	X	-	-	-	-	X	P	X
_POSIX2_C_DEV	X	X	X	X	-	-	-	-	X	-	X
_POSIX2_CHAR_TERM	-	-	-	X	-	-	-	-	X	-	X
_POSIX2_FORT_DEV	-	-	-	-	-	-	-	-	o	-	o
_POSIX2_FORT_RUN	-	-	-	X	-	-	-	-	o	-	o
_POSIX2_LOCALEDEF	-	-	-	-	-	-	-	-	X	-	X
_POSIX2_SW_DEV	X	X	X	X	-	-	-	-	o	-	o
_POSIX2_UPE	-	-	-	X	-	-	-	-	X	-	X

Portability: PSE51 _POSIX_SINGLE_PROCESS

- Percent Complete: 33%
- Remaining: 2
  - *sysconf()* - incredibly useful. There is also RFC #56670
  - *uname()* - also kind of useful
- Exceptions:
  - *confstr(), environ, getenv(), setenv(), unsetenv()*

Portability: PSE51 _POSIX_SIGNALS

- Percent Complete: 12.5%
- Remaining: 7
  - *sigaction( ) ? , sigaddset( ), sigdelset( ), sigemptyset( ), sigfillset( ), sigismember( ), sigpending( ),*
- Exceptions:
  - *alarm( ), kill( ), pause( ), raise( ), signal( ), sigprocmask( ), sigsuspend( ), sigwait( )*
- Notes:
  - With some effort, it is even possible to create a per-thread signal handler.
  - The only thing that cannot be supported with PSE51 are signals that implicitly affect the whole process, since there is only 1 process
  - There may be some lower-layer modifications necessary to create a cancellation point

Portability: PSE51 _POSIX_THREADS_BASE

- Percent Complete: 76%
- Remaining: 12
  - *pthread_atfork()*
  - *pthread_barrierattr_destroy()*, *pthread_barrierattr_init()*
  - *pthread_barrierattr_getpshared()*, *pthread_barrierattr_setpshared()*
  - *pthread_cleanup_pop()*, *pthread_cleanup_push()*
  - *pthread_equal()*
  - *pthread_kill()*
  - *pthread_sigmask()*
  - *pthread_setcancelstate()*, *pthread_testcancel()*
- Exceptions:
  - None!

Option Requirements: _POSIX_CLOCK_SELECTION

- Percent Complete: 0%
- Remaining: 3
  - *pthread_condattr_getclock()*, *pthread_condattr_setclock()*, *clock_nanosleep()*,
- Exceptions:
  - None
- Notes:
  - Implies `_POSIX_TIMERS`

Option Requirements: _POSIX_SHARED_MEMORY_OBJECTS

- Percent Complete: 0%
- Remaining: 4
  - mmap()
  - munmap()
  - shm_open()
  - shm_unlink()
- Exceptions:
  - None!

Option Requirements: _POSIX_CPUTIME

- Percent Complete: 0%
- Remaining: 1
    - clock_getcpuclockid()
- Exceptions:
    - None!
- Notes:
    - Implies `_POSIX_TIMERS`
    - Implications: per-cpu counters!
        - Devicetree boolean property: *cpu-counter*

Option Requirements: _POSIX_TIMERS

- Percent Complete: 77%
- Remaining: 2
    - *clock_getres()*
    - *timer_getoverrun()*
- Exceptions:
    - *aio_suspend()* (not in the 1003.1-2017 spec)
- Notes:
    - This should be done at the Zephyr layer. POSIX becomes a thin wrapper around the Zephyr (#19030, #40099), *CLOCK_MONOTONIC* -> *K_CLOCK_MONOTONIC*, *k_nanosleep()*, etc
    - Highly relevant for e.g.. Time Synchronized Channel Hopping (TSCH) in 802.15.4
    - Official Zephyr chosen nodes in Devicetree
        - *zephyr,real_time_clock = &rtc0*
        - *zephyr,monotonic_clock = &cpu_counter_0*

03 How it’s going

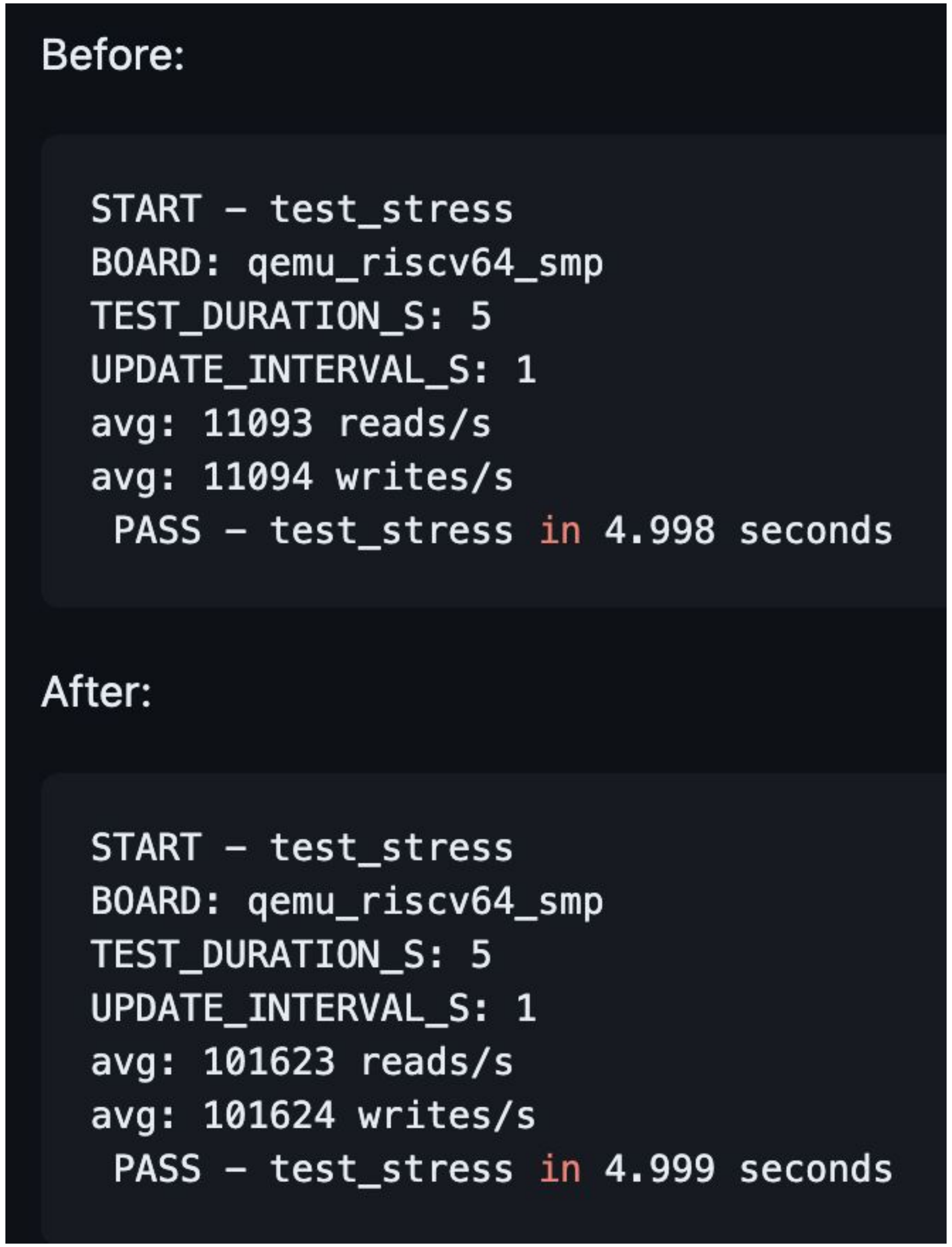
Since Becoming Maintainer..

After 6 months, 9/16 Tasks complete 🙌 (See [2023-01 Slides from Architecture Meeting])

🚀

Since Becoming Maintainer..

- After 10 months, improved ***eventfd_read()*** and ***eventfd_write()*** performance by ***10x****



Before:

```
START - test_stress
BOARD: qemu_riscv64_smp
TEST_DURATION_S: 5
UPDATE_INTERVAL_S: 1
avg: 11093 reads/s
avg: 11094 writes/s
 PASS - test_stress in 4.998 seconds
```

After:

```
START - test_stress
BOARD: qemu_riscv64_smp
TEST_DURATION_S: 5
UPDATE_INTERVAL_S: 1
avg: 101623 reads/s
avg: 101624 writes/s
 PASS - test_stress in 4.999 seconds
```

* with only 1 teensy bug

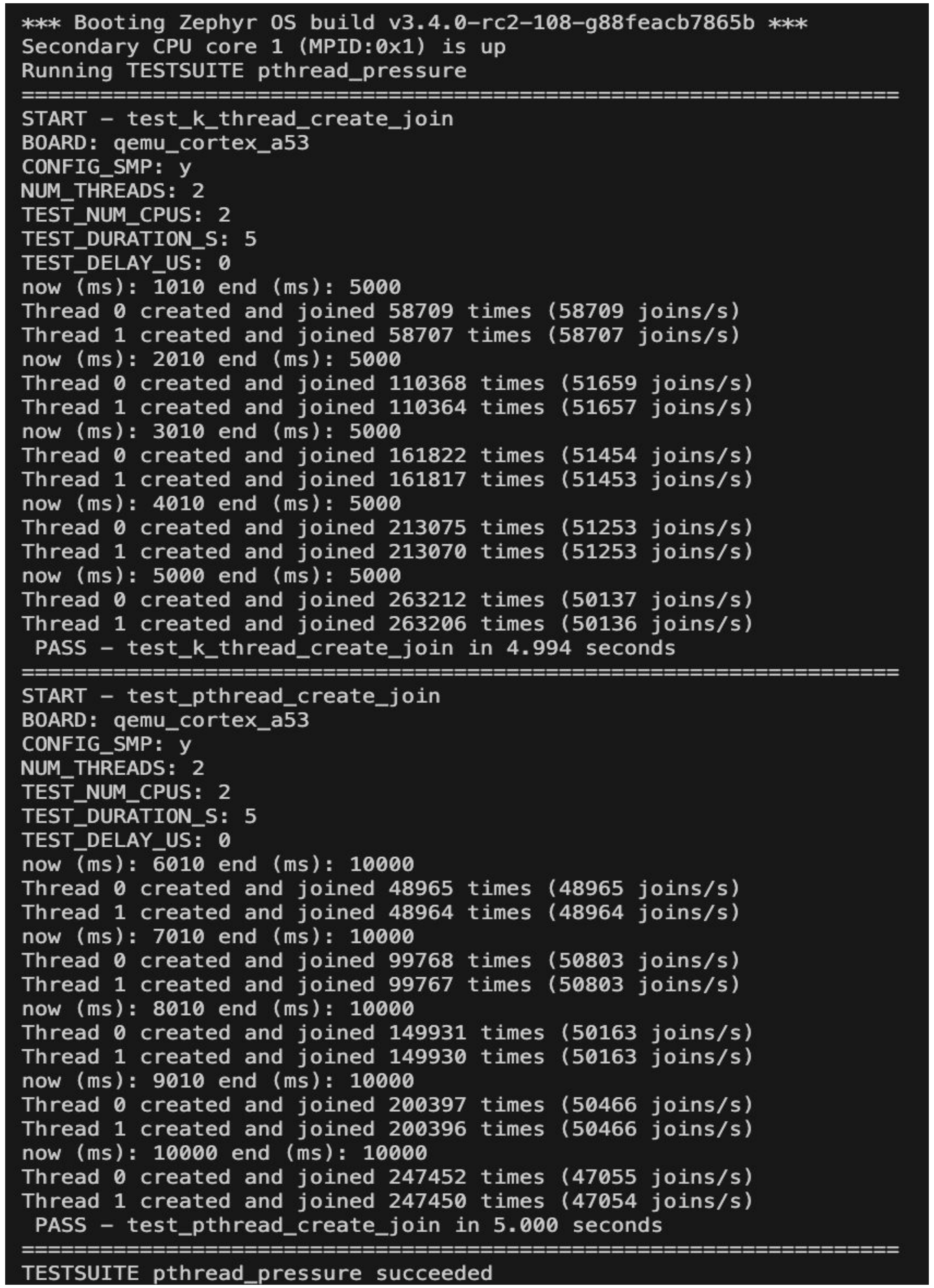
Since Becoming Maintainer..

- After 11 months, improved ***pthread_create()*** and ***pthread_join()*** performance by ***???x****†

🚀 🥳 🎉

* also uncovered k_thread race condition

† test would crash prior to the change, so unable to measure true perf



```
*** Booting Zephyr OS build v3.4.0-rc2-108-g88feacb7865b ***
Secondary CPU core 1 (MPID:0x1) is up
Running TESTSUITE pthread_pressure
===================================================================
START - test_k_thread_create_join
BOARD: qemu_cortex_a53
CONFIG_SMP: y
NUM_THREADS: 2
TEST_NUM_CPUS: 2
TEST_DURATION_S: 5
TEST_DELAY_US: 0
now (ms): 1010 end (ms): 5000
Thread 0 created and joined 58709 times (58709 joins/s)
Thread 1 created and joined 58707 times (58707 joins/s)
now (ms): 2010 end (ms): 5000
Thread 0 created and joined 110368 times (51659 joins/s)
Thread 1 created and joined 110364 times (51657 joins/s)
now (ms): 3010 end (ms): 5000
Thread 0 created and joined 161822 times (51454 joins/s)
Thread 1 created and joined 161817 times (51453 joins/s)
now (ms): 4010 end (ms): 5000
Thread 0 created and joined 213075 times (51253 joins/s)
Thread 1 created and joined 213070 times (51253 joins/s)
now (ms): 5000 end (ms): 5000
Thread 0 created and joined 263212 times (50137 joins/s)
Thread 1 created and joined 263206 times (50136 joins/s)
 PASS - test_k_thread_create_join in 4.994 seconds
===================================================================
START - test_pthread_create_join
BOARD: qemu_cortex_a53
CONFIG_SMP: y
NUM_THREADS: 2
TEST_NUM_CPUS: 2
TEST_DURATION_S: 5
TEST_DELAY_US: 0
now (ms): 6010 end (ms): 10000
Thread 0 created and joined 48965 times (48965 joins/s)
Thread 1 created and joined 48964 times (48964 joins/s)
now (ms): 7010 end (ms): 10000
Thread 0 created and joined 99768 times (50803 joins/s)
Thread 1 created and joined 99767 times (50803 joins/s)
now (ms): 8010 end (ms): 10000
Thread 0 created and joined 149931 times (50163 joins/s)
Thread 1 created and joined 149930 times (50163 joins/s)
now (ms): 9010 end (ms): 10000
Thread 0 created and joined 200397 times (50466 joins/s)
Thread 1 created and joined 200396 times (50466 joins/s)
now (ms): 10000 end (ms): 10000
Thread 0 created and joined 247452 times (47055 joins/s)
Thread 1 created and joined 247450 times (47054 joins/s)
 PASS - test_pthread_create_join in 5.000 seconds
===================================================================
TESTSUITE pthread_pressure succeeded
```

Since Becoming Maintainer..

- After 12 months, 17/54 Tasks complete?* (See RFC #51211)

Since Becoming Maintainer..

- After 12 months, ~~17~~ 38/54 = 70% Tasks complete*

* We just broke down the remaining Tasks into smaller, more manageable ones

04 What next?

Dynamic Thread Stacks

- Dynamic Zephyr thread stacks are **nearly complete!!1!** 🚀 🥳 🎉
- This is **MASSIVE** - will not only facilitate proper pthreads but (as a result)
  - ISO C11 threads (*<threads.h>*)
  - ISO C++11 *std::thread*, *std::mutex*, *std::condition_variable*, ..
  - Other languages? 🦀🐍
- Simple API for both pool and heap allocation (so OK for safety critical)
  - *k_thread_stack_allocate(..)*
  - *k_thread_stack_free(..)*
- Incredible last-minute collaboration with Intel 🙏
- Coming soon..

CONFIG_ARCH_POSIX && CONFIG_POSIX_API

- If all goes well, the ability to build / run / test Zephyr's POSIX API on *native_posix* and *native_posix_64* 🚀 🥳 🎉
- Actually, *native_sim*, and *native_sim_64* (new boards in the process of being added)
- Recent work presented at the Arch meeting by the POSIX Arch Maintainer (now at Nordic) 🙏

Pick the low-hanging fruit

- The great news about breaking down the remaining tasks into smaller, more manageable tasks, is that a large percentage of them are trivial to implement 🚀 🥳 🎉
- Great opportunity for new contributors to Zephyr!!
- **POSIX Collaborators welcome!** Existing contributors to Zephyr could get their name into 👉 MAINTAINERS.yml 👈

List of missing POSIX functions with trivial implementations

sysconf(), *uname()*, *sigaddset()*, *sigdelset()*, *sigemptyset()*, *sigfillset()*, *sigismember()*, *sigpending()*, *pthread_atfork()*, *pthread_barrierattr_destroy()*, *pthread_barrierattr_init()*, *pthread_barrierattr_getpshared()*, *pthread_barrierattr_setpshared()*, *pthread_cleanup_pop()*, *pthread_cleanup_push()*, *pthread_equal()*, *pthread_kill()*, *pthread_sigmask()*, *pthread_condattr_getclock()*, *pthread_condattr_setclock()*, *clock_getcpuclockid()*

Questions? / Feedback

Meta