



OpenThread

Thread Technology Workshop

Jonathan Hui

September 14th, 2017

Building a more
thoughtful home



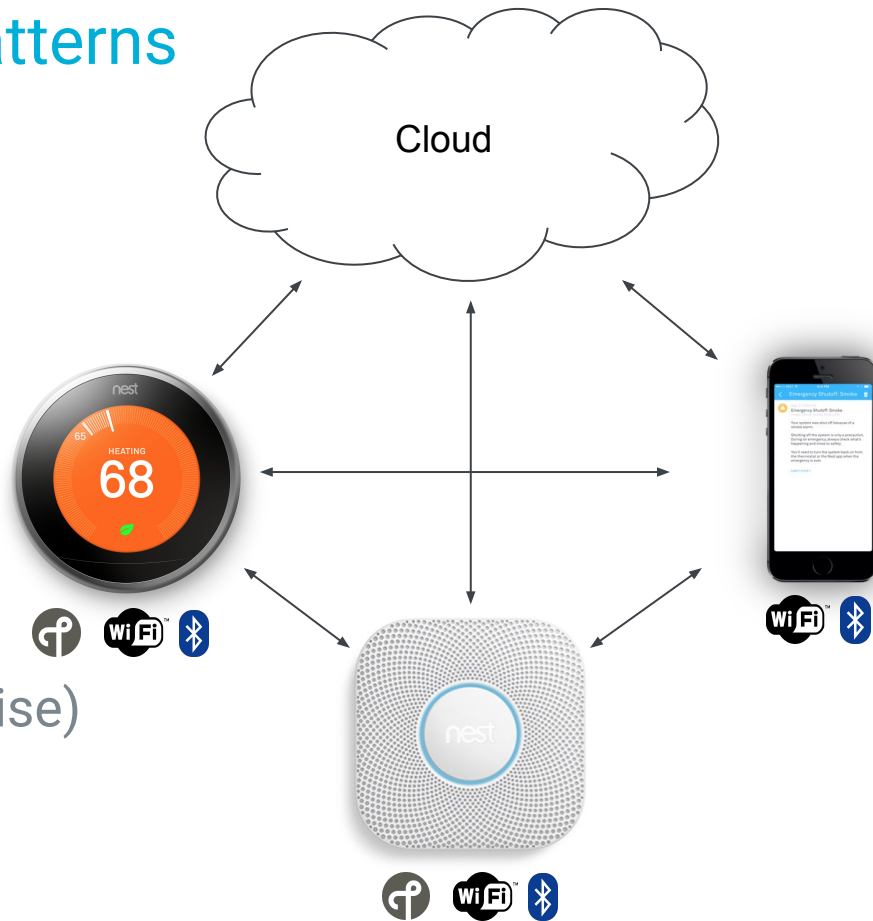
Diverse Communication Patterns

Multiple Interfaces

- WiFi
- Thread
- BLE

Communicates with

- Cloud (off premise)
- Thermostat (on premise)
- Mobile Phone (off or on premise)



OpenThread

OpenThread

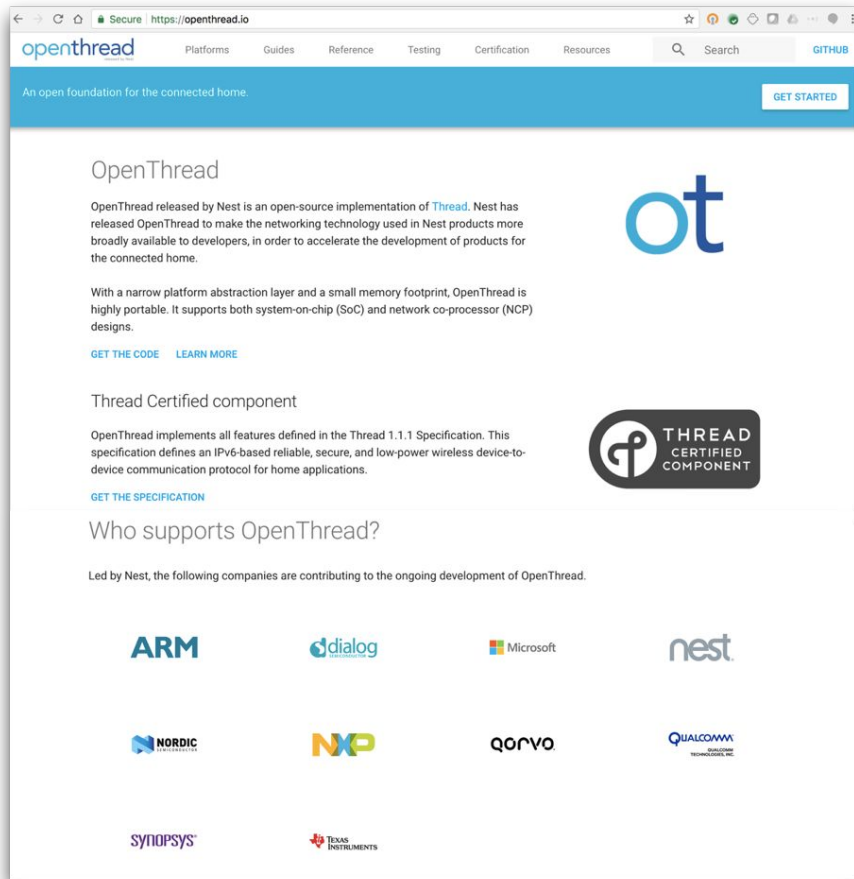
openthread.io

Open-source Thread implementation

- Thread Certified Component
- BSD-3 license
- Supported by multiple vendors and community
- Used in Nest products



GitHub



Why Open Source?

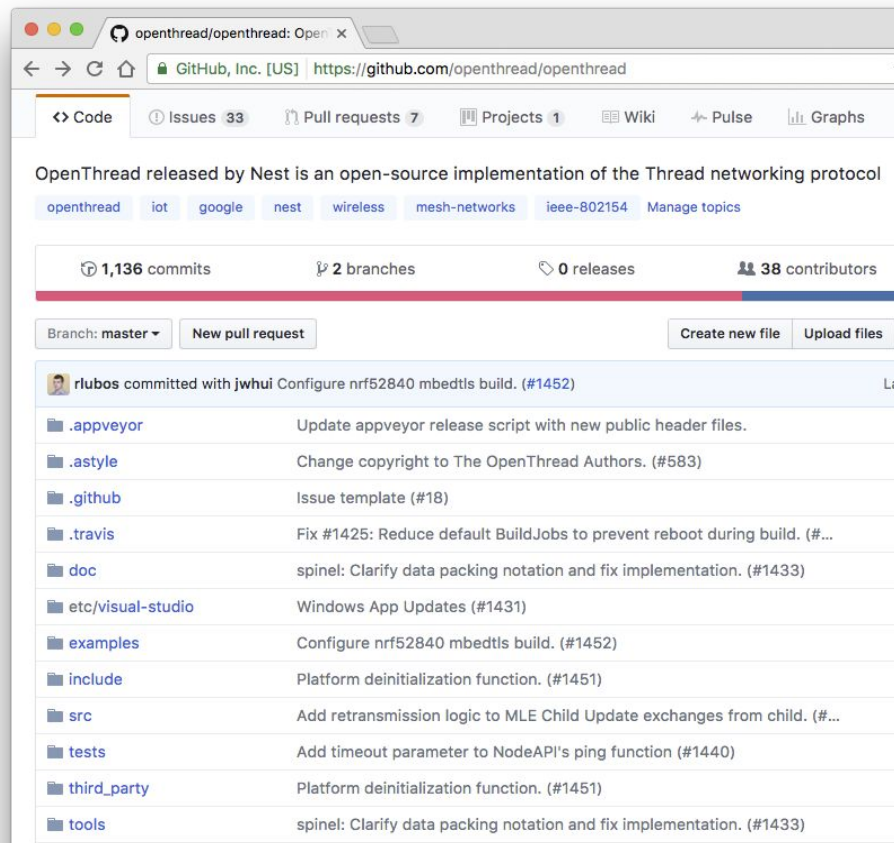
github.com/openthread/openthread

Full and unencumbered source access

- Debugging
- Flexible hardware selection
- Minimal business constraints
- Vendor and community support

Flexible Architecture

- Single code base for all products
- Hardware and OS independent
- Build SoC and NCP-based products
- Optimize memory and performance to platform



OpenThread Core Library

Thread 1.1.1 Complete

Portable C/C++ library (C99 and C++03)

- Platform agnostic (narrow platform abstraction)
- OS agnostic (bare-metal, FreeRTOS, Linux, Windows)

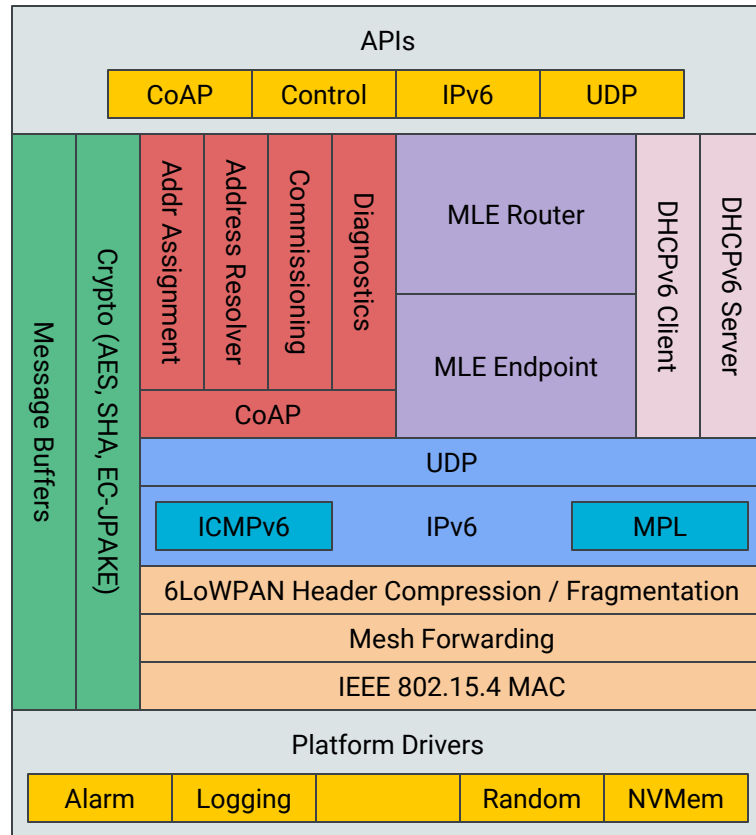
Flexible system architecture

- Single-chip and Network Co-Processor support
- Flexible build configurations (e.g. FTD vs. MTD)

MTD 71 KiB Code 10 KiB RAM

FTD 109 KiB Code 16 KiB RAM

mbedTLS 73 KiB Code 6 KiB RAM



Rich C API

include/openthread

Control

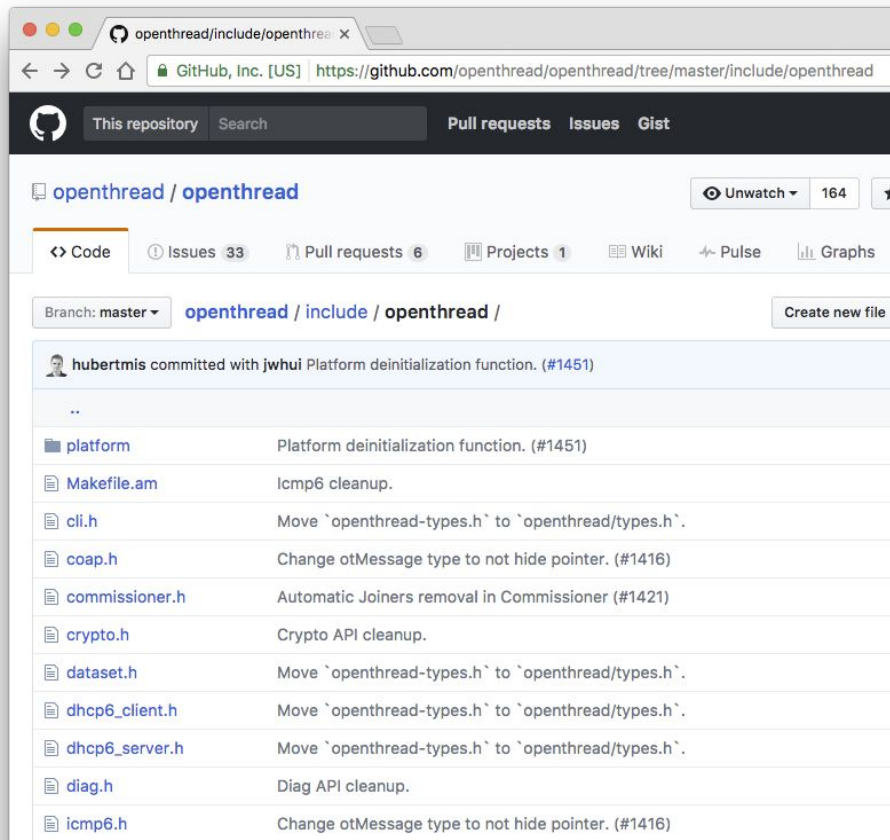
- Thread (MLE, Commissioning, Border Router, ...)
- IPv6 Protocol

Data

- UDP Sockets
- IPv6 Raw

Optional

- CoAP Client and Server
- DHCPv6 Client and Server
- DNSv6 Client
- Factory Diagnostics
- Network Co-Processor (NCP)
- Command Line Interface (CLI)

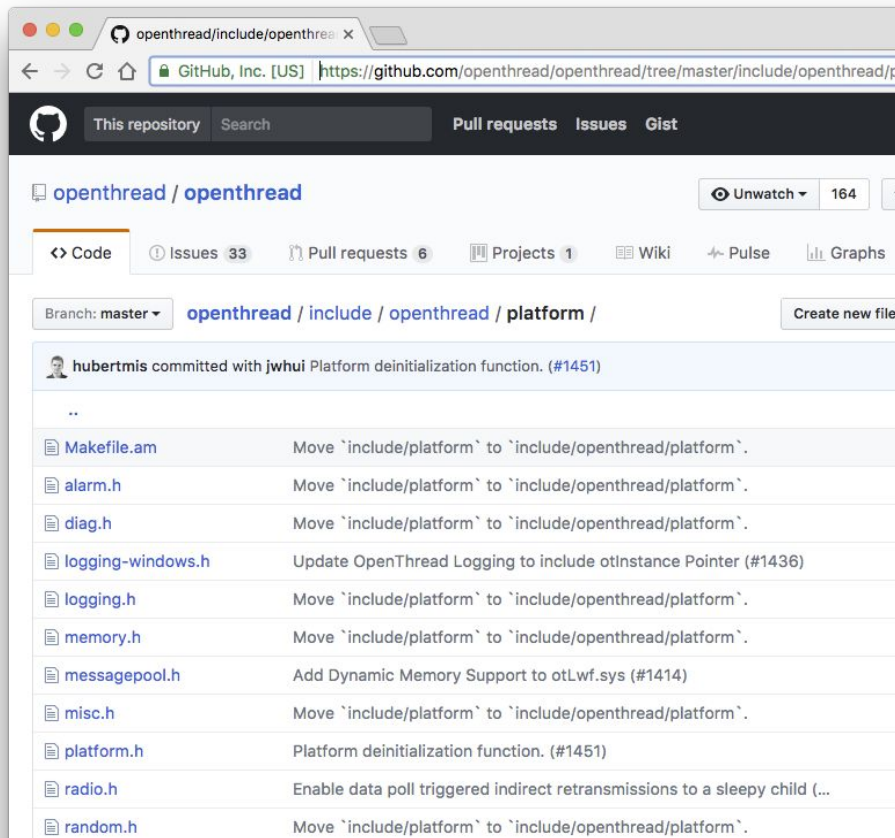


Narrow Platform Abstraction

include/openthread/platform

Required platform services

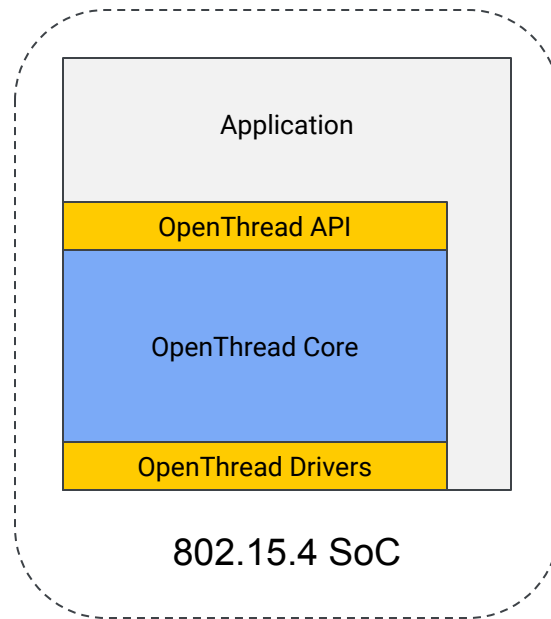
- Alarm
- Radio
- Random
- Settings (non-volatile)



Single-Chip, Thread-Only

Highly integrated low-cost, low-power

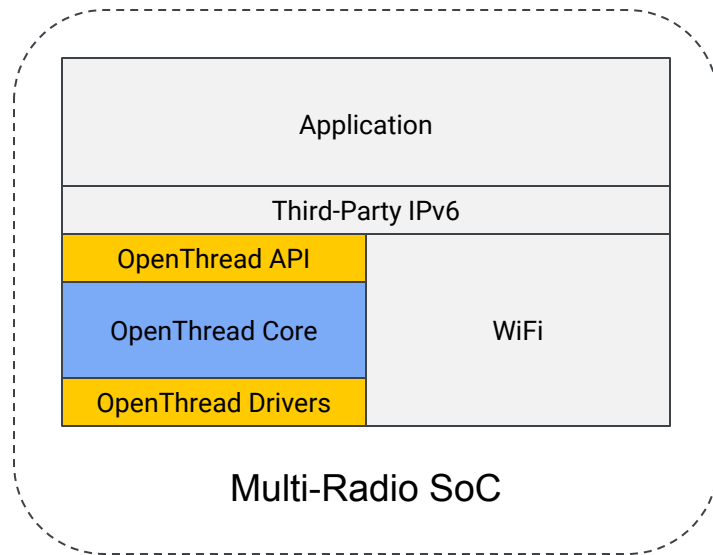
- Application and OpenThread on same core
- Application directly calls OpenThread API
- Utilize OpenThread's CoAP/UDP/IPv6 stack



Single-Chip, Multiple-Interface

OpenThread as a link technology

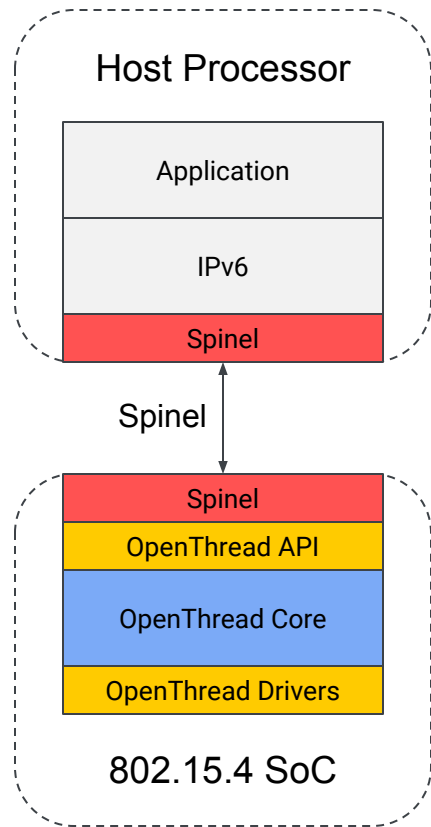
- Application and OpenThread on same core
- OpenThread as a link technology to IPv6 stack
- Input/output raw IPv6 datagrams



Network Co-Processor (NCP)

Offload Thread functions to an SoC

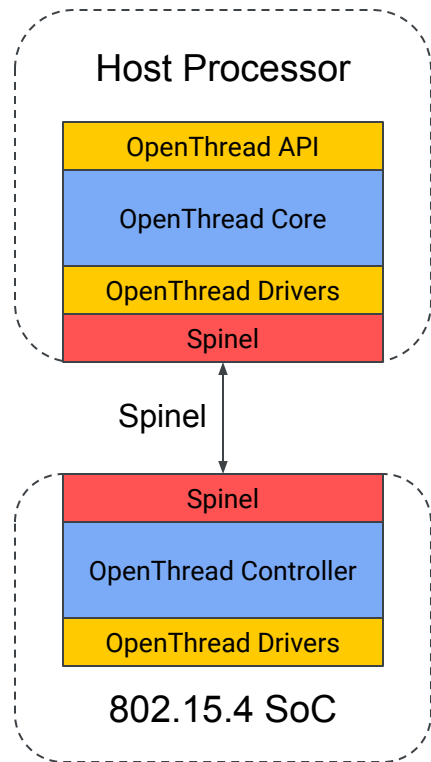
- Support designs with a more-capable (but higher power) host processor
- Thread offload allows host processor to sleep while maintaining Thread network
- Spinel for control protocol



Host / 802.15.4 Controller

Offload 802.15.4 functions to an SoC

- Support designs with a more-capable host processor
- OpenThread utilizing host processor resources
- Spinel control protocol



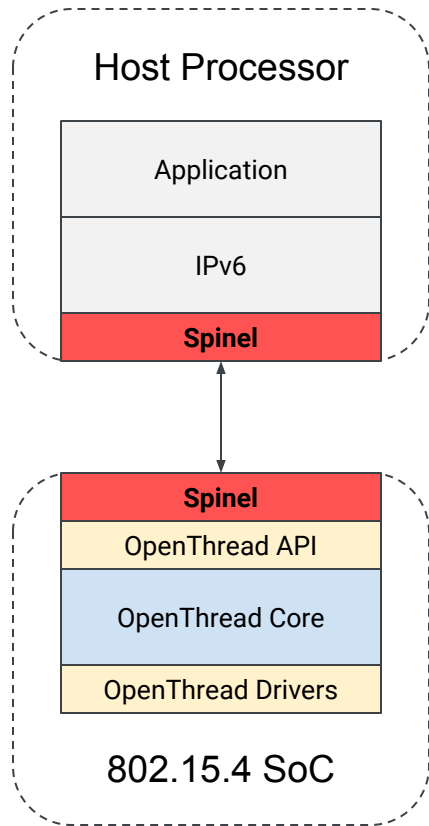
OpenThread Projects

NCP Control Protocol: Spinel

goo.gl/xa9Coe

General-purpose protocol for using a NCP

- Useful when network-layer stack is partially off-chip
- Provides a rich, property-based API
- Specializations for 802.15.4 and Thread
- Supports multiple logical network interfaces
- Can be extended while remaining backward compatible
- Transport framing protocol specified for UART and SPI

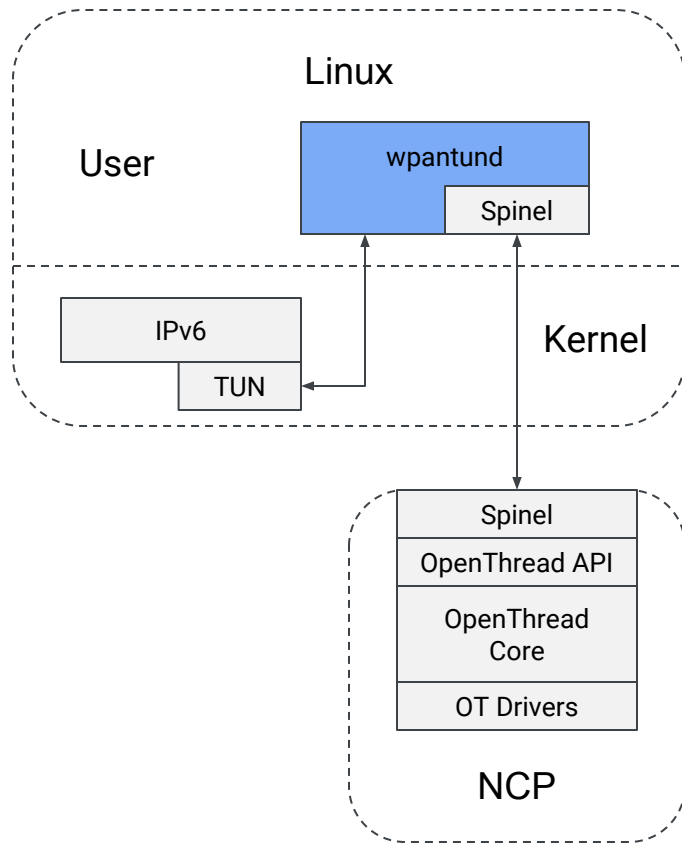


wpantund Project

github.com/openthread/wpantund

User-space NCP network interface driver

- “pppd for NCPs”
- IPv6 connectivity to Thread on Linux and macOS
- Manages NCP power, firmware updates, scanning, joining, forming, and network attach/detach
- Convenient command-line interface: wpanctl
- DBus-based management API
- Uses Spinel protocol for NCP management



Border Router Project

github.com/openthread/borderrouter

End-to-end IPv4/6 Connectivity

- Stateful NAT64 for IPv4 connectivity

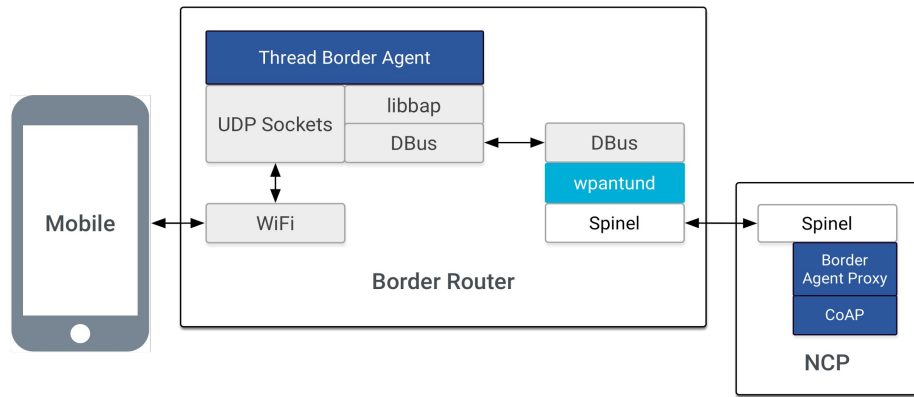
External Thread Commissioner

- Thread Border Agent

Web UI for Config and Management

Hardware

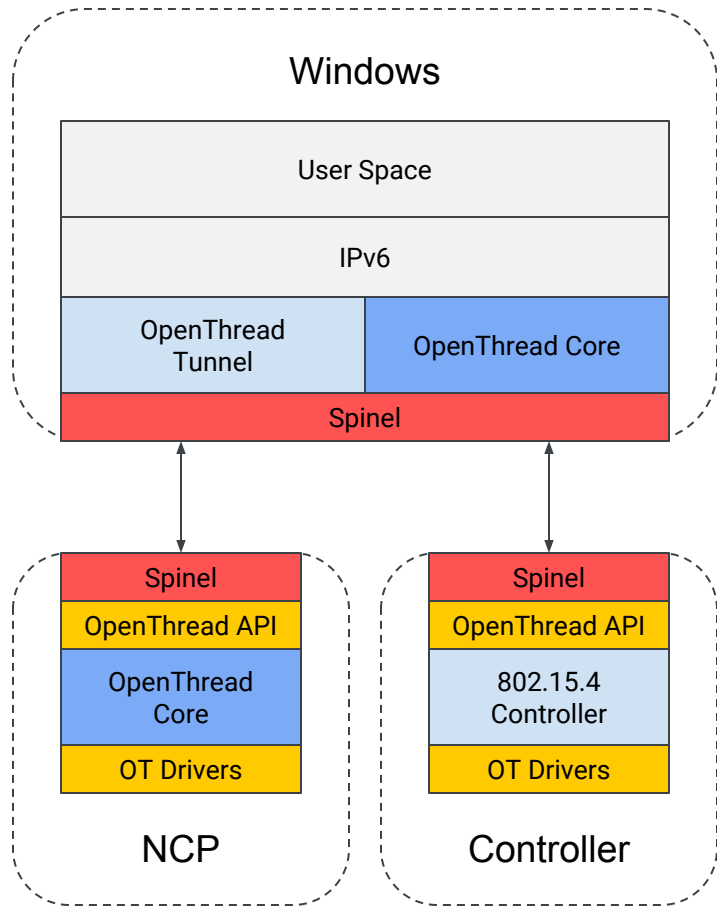
- Raspberry Pi 3B (BeagleBone coming soon)
- OpenThread Network Co-Processor (NCP)



Windows 10

github.com/openthread/openthread/wiki/Building-on-Windows

- Native Windows 10 and Windows 10 IoT support
- Contributed and maintained by Microsoft
- Two supported architectures
 - OpenThread on NCP (Windows driver in tunnel mode)
 - OpenThread on Windows (NCP in controller mode)
- wpantund-like solution for Windows platform



OpenThread Development

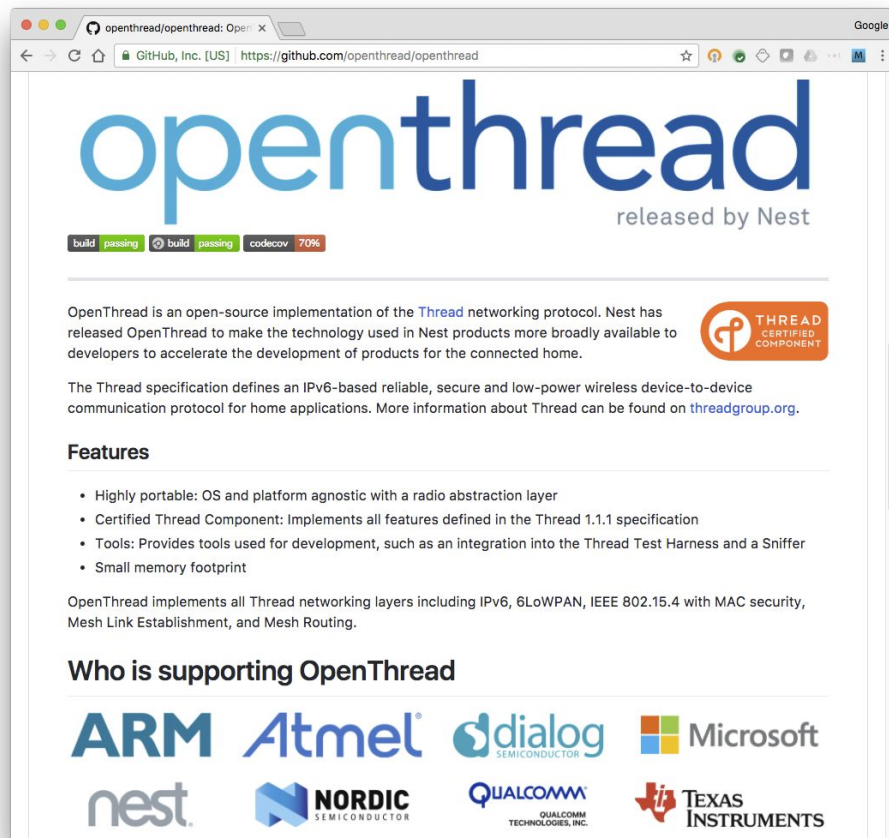
Source Code

github.com/openthread/openthread

Freely available on GitHub

- BSD-3 license
- Supported by multiple vendors and community

GitHub



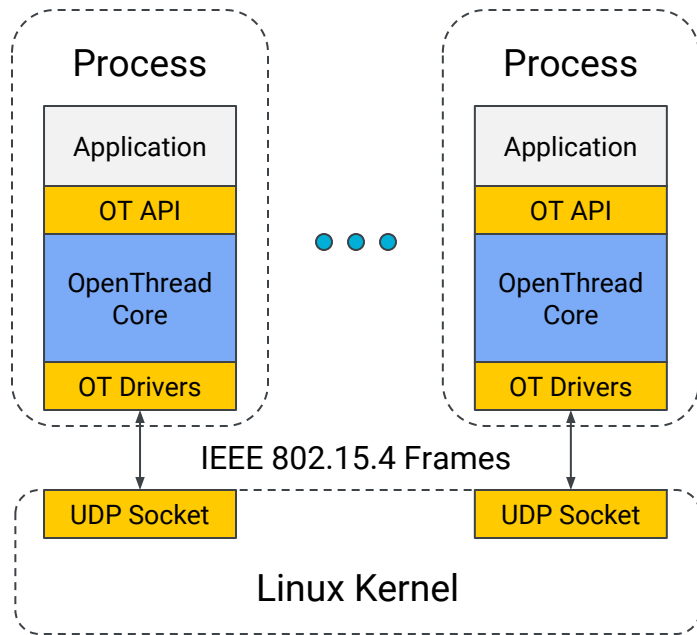
Development Kit Availability



POSIX Emulation

Emulate OpenThread device as POSIX process

- One process per device
- IEEE 802.15.4 radio driver on top of UDP sockets
- Simulate and test OpenThread networks without hardware
- Utilized by OpenThread Continuous Integration



Toolchain Agnostic

GNU Toolchain

- Support open and free development
- GNU make, GCC, GDB, binutils, autotools, etc.

Proprietary Toolchain Support

- IAR
- Visual Studio
- Other vendor-specific toolchains



Flexible Build Configuration

Optimize to your application needs

Feature Options

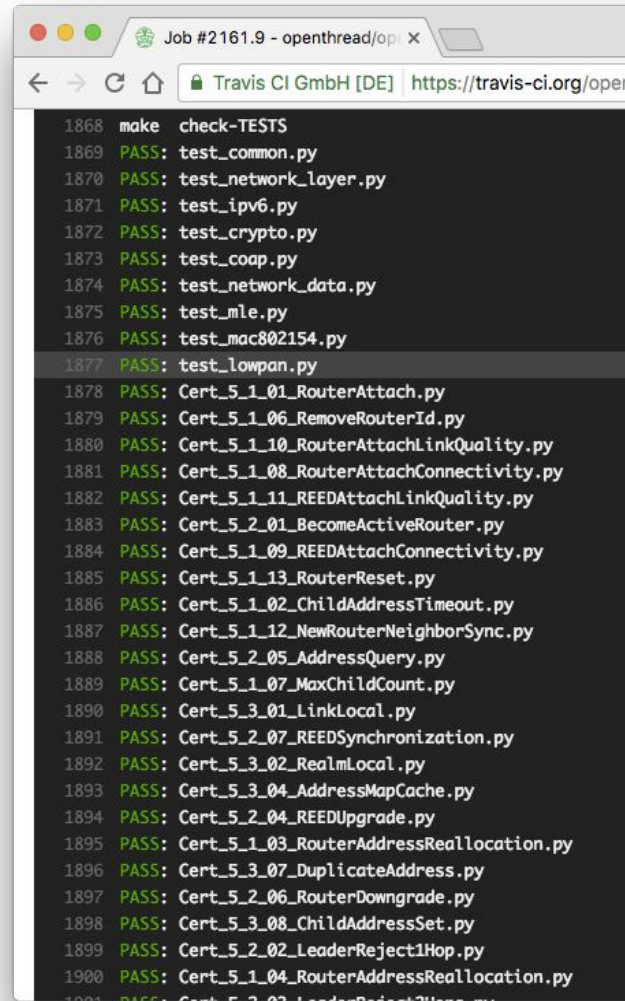
- Minimal and Full Thread Device Builds
- Network Co-Processor (NCP)
- Command Line Interface (CLI)
- Border Router
- CoAP Client and Server for Application
- DHCPv6 Client and Server
- DNSv6 Client
- Raw Link for 802.15.4 Controller
- Logging
- Factory Diagnostics

Configuration Parameters

- Message buffer pool size
- Maximum number of children
- Maximum number of IPv6 addresses
- Log components and level
- Protocol options for CoAP, DNS, MLE, etc.

Automated Test Infrastructure

- **Static Analysis** - Clang, Coverity, Visual Studio
- **Unit Tests** - test modules in isolation
- **Functional Testing** - test protocol behavior in emulation
- **Address Sanitizer** - compiler-instrumented memory error checking
- **libFuzzer via OSS-Fuzz** - in-process, coverage-guided, evolutionary fuzzing
- **Code Coverage Analysis** - measure degree of test coverage

A screenshot of a web browser showing a Travis CI build log for the OpenThread project. The browser's address bar shows the URL 'https://travis-ci.org/open'. The log displays a series of test results, each preceded by a line number and the word 'PASS:'. The tests include 'test_common.py', 'test_network_layer.py', 'test_ipv6.py', 'test_crypto.py', 'test_coap.py', 'test_network_data.py', 'test_mle.py', 'test_mac802154.py', 'test_lowpan.py', and a series of 'Cert' tests such as 'Cert_5_1_01_RouterAttach.py', 'Cert_5_1_06_RemoveRouterId.py', 'Cert_5_1_10_RouterAttachLinkQuality.py', 'Cert_5_1_08_RouterAttachConnectivity.py', 'Cert_5_1_11_REEDAttachLinkQuality.py', 'Cert_5_2_01_BecomeActiveRouter.py', 'Cert_5_1_09_REEDAttachConnectivity.py', 'Cert_5_1_13_RouterReset.py', 'Cert_5_1_02_ChildAddressTimeout.py', 'Cert_5_1_12_NewRouterNeighborSync.py', 'Cert_5_2_05_AddressQuery.py', 'Cert_5_1_07_MaxChildCount.py', 'Cert_5_3_01_LinkLocal.py', 'Cert_5_2_07_REEDSynchronization.py', 'Cert_5_3_02_RealmLocal.py', 'Cert_5_3_04_AddressMapCache.py', 'Cert_5_2_04_REEDUpgrade.py', 'Cert_5_1_03_RouterAddressReallocation.py', 'Cert_5_3_07_DuplicateAddress.py', 'Cert_5_2_06_RouterDowngrade.py', 'Cert_5_3_08_ChildAddressSet.py', 'Cert_5_2_02_LeaderReject1Hop.py', and 'Cert_5_1_04_RouterAddressReallocation.py'. The log is displayed in a dark-themed editor with green text for 'PASS' and red for 'make' and 'check-TESTS'.

```
1868 make check-TESTS
1869 PASS: test_common.py
1870 PASS: test_network_layer.py
1871 PASS: test_ipv6.py
1872 PASS: test_crypto.py
1873 PASS: test_coap.py
1874 PASS: test_network_data.py
1875 PASS: test_mle.py
1876 PASS: test_mac802154.py
1877 PASS: test_lowpan.py
1878 PASS: Cert_5_1_01_RouterAttach.py
1879 PASS: Cert_5_1_06_RemoveRouterId.py
1880 PASS: Cert_5_1_10_RouterAttachLinkQuality.py
1881 PASS: Cert_5_1_08_RouterAttachConnectivity.py
1882 PASS: Cert_5_1_11_REEDAttachLinkQuality.py
1883 PASS: Cert_5_2_01_BecomeActiveRouter.py
1884 PASS: Cert_5_1_09_REEDAttachConnectivity.py
1885 PASS: Cert_5_1_13_RouterReset.py
1886 PASS: Cert_5_1_02_ChildAddressTimeout.py
1887 PASS: Cert_5_1_12_NewRouterNeighborSync.py
1888 PASS: Cert_5_2_05_AddressQuery.py
1889 PASS: Cert_5_1_07_MaxChildCount.py
1890 PASS: Cert_5_3_01_LinkLocal.py
1891 PASS: Cert_5_2_07_REEDSynchronization.py
1892 PASS: Cert_5_3_02_RealmLocal.py
1893 PASS: Cert_5_3_04_AddressMapCache.py
1894 PASS: Cert_5_2_04_REEDUpgrade.py
1895 PASS: Cert_5_1_03_RouterAddressReallocation.py
1896 PASS: Cert_5_3_07_DuplicateAddress.py
1897 PASS: Cert_5_2_06_RouterDowngrade.py
1898 PASS: Cert_5_3_08_ChildAddressSet.py
1899 PASS: Cert_5_2_02_LeaderReject1Hop.py
1900 PASS: Cert_5_1_04_RouterAddressReallocation.py
```

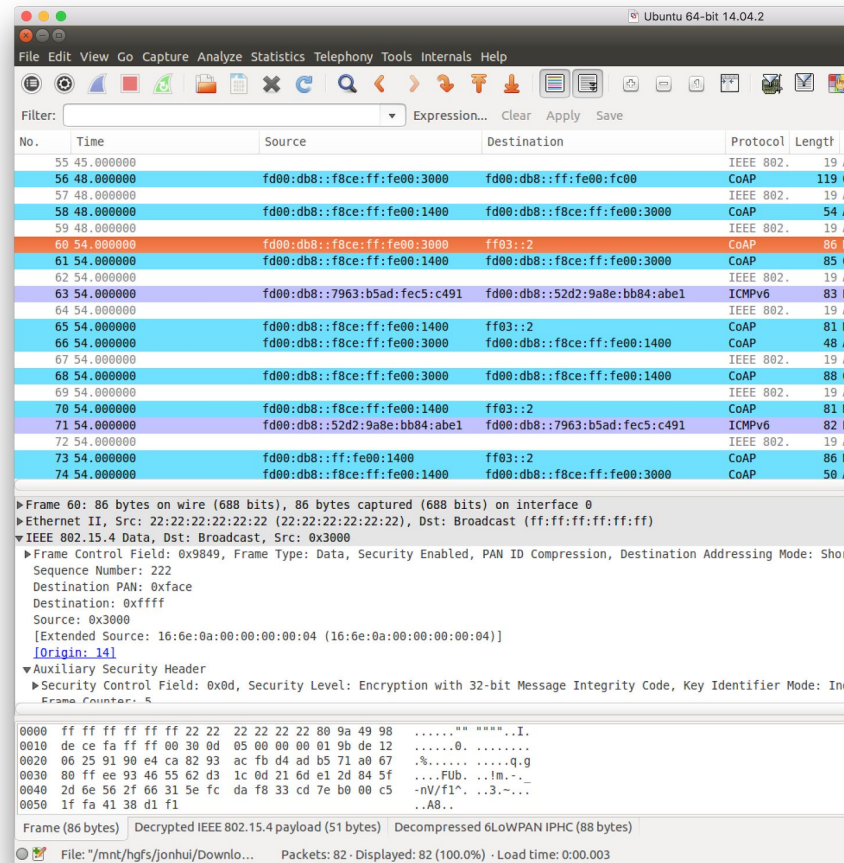
OpenThread Sniffer

NCP build has sniffer support built-in

- Single image for network operation and debugging
- Exposed via Spinel protocol
- Monitor mode (capture packets during operation)
- Promiscuous mode (dedicated sniffer)

Host-side support

- wpantund
- Stand-alone python tool (tools/spinel-cli/sniffer.py)
- Output pcap file format
- Pipe directly to Wireshark



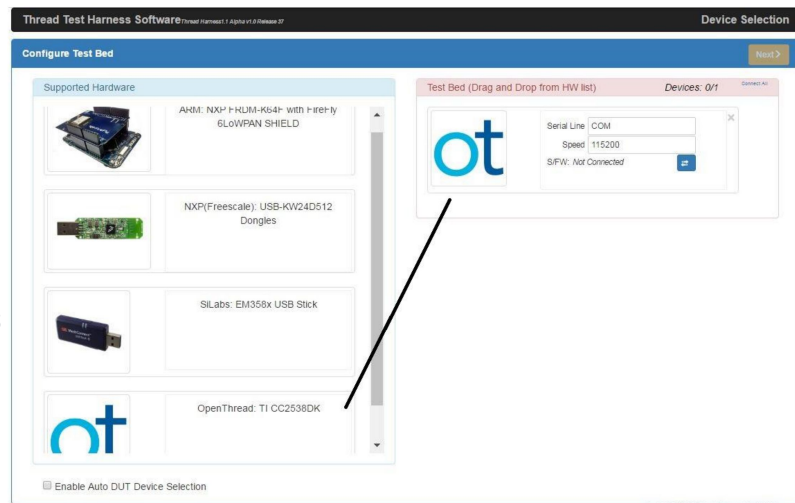
Thread Certification Tools

Thread Harness Automation

- Selenium-based
- tools/harness-automation

Thread Harness THCI for OpenThread

- Allows Thread Harness to control OpenThread directly
- Used to control OpenThread reference and DUT devices
- Maps to OpenThread CLI module
- tools/harness-thci



Continuous Integration

Travis CI



Travis CI

- Free Linux and MacOS-based CI cloud service
- GCC (native and arm) 4, 5, and 6
- Clang static analysis
- Unit and functional tests
- SoC and NCP

AppVeyor



- Free Windows-based cloud service
- Visual Studio
- Unit tests
- Windows-kernel and NCP modes

openthread / openthread **build passing**

Current Branches Build History Pull Requests

✓ **master** Separate build output based on target. (#1482) **passing** #4569 passed

Commit 3872825
Compare 0b66335..3872825
Branch master

Jonathan Hui authored GitHub committed

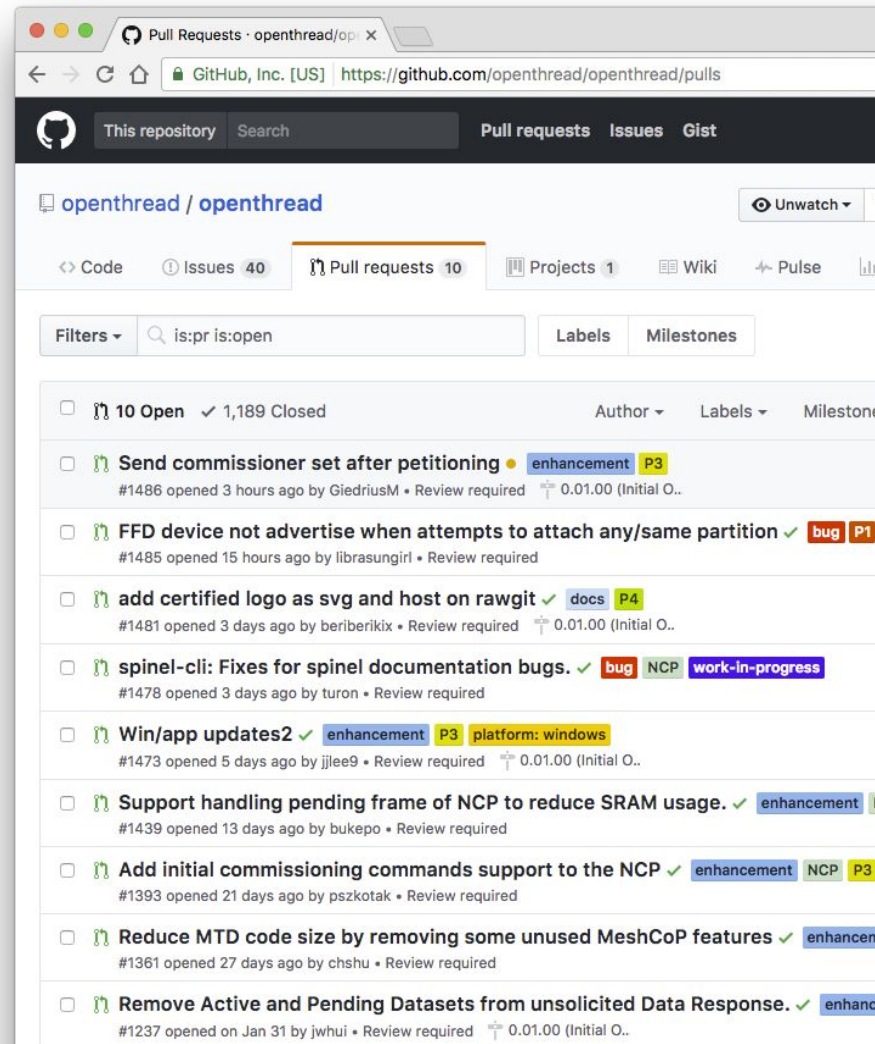
Ran for 40 min 30 sec
Total time 1 hr 28 min 3 sec
about 2 hours ago

Build Jobs

✓ # 4569.1	</> no language set BUILD_TARGET="pretty-check"	1 min 7 s
✓ # 4569.2	</> Compiler: clang BUILD_TARGET="scan-build" CC="clang" CXX="clang++"	5 min 17 s
✓ # 4569.3	</> Compiler: gcc BUILD_TARGET="arm-gcc49"	4 min 14 s
✓ # 4569.4	</> Compiler: gcc BUILD_TARGET="arm-gcc54"	4 min 44 s
✓ # 4569.5	</> Compiler: clang BUILD_TARGET="posix" CC="clang" CXX="clang++"	1 min 23 s
✓ # 4569.6	</> Compiler: gcc BUILD_TARGET="posix" CC="gcc" CXX="g++"	1 min 17 s
✓ # 4569.7	</> Compiler: gcc BUILD_TARGET="posix" CC="gcc-5" CXX="g++-5"	2 min 39 s
✓ # 4569.8	</> Compiler: gcc BUILD_TARGET="posix" CC="gcc-6" CXX="g++-6"	2 min 40 s
✓ # 4569.9	</> Compiler: clang BUILD_TARGET="posix-distcheck" VERBOSE=1	25 min 3 s
✓ # 4569.10	</> Compiler: gcc BUILD_TARGET="posix-32-bit" VERBOSE=1	23 min 5 s
✓ # 4569.11	</> Compiler: gcc BUILD_TARGET="posix-ncp-spi" VERBOSE=1	1 min 16 s
✓ # 4569.12	</> Compiler: gcc BUILD_TARGET="posix-ncp" VERBOSE=1	13 min 5 s

Anyone can contribute!

- Discuss
 - a. openthread-users@googlegroups.com
 - b. Stack Overflow with [\[openthread\]](#) tag
- Find a bug?
 - a. File a [GitHub Issue](#)
- New Feature?
 - a. Propose in a [GitHub Issue](#)
 - b. Implement and push to GitHub Fork
 - c. Validate continuous integration results
 - d. Submit to upstream in a [GitHub Pull Request](#)



Where to go from here

View the source

- github.com/openthread/openthread
- github.com/openthread/borderrouter
- github.com/openthread/wpantund

Play in simulation

- Codelab at codelabs.developers.google.com/codelabs/openthread-simulation

Play on hardware

- Dialog, Nordic, NXP, Qorvo, Silicon Labs, Synopsys, Texas Instruments

Nordic Thread SDK

