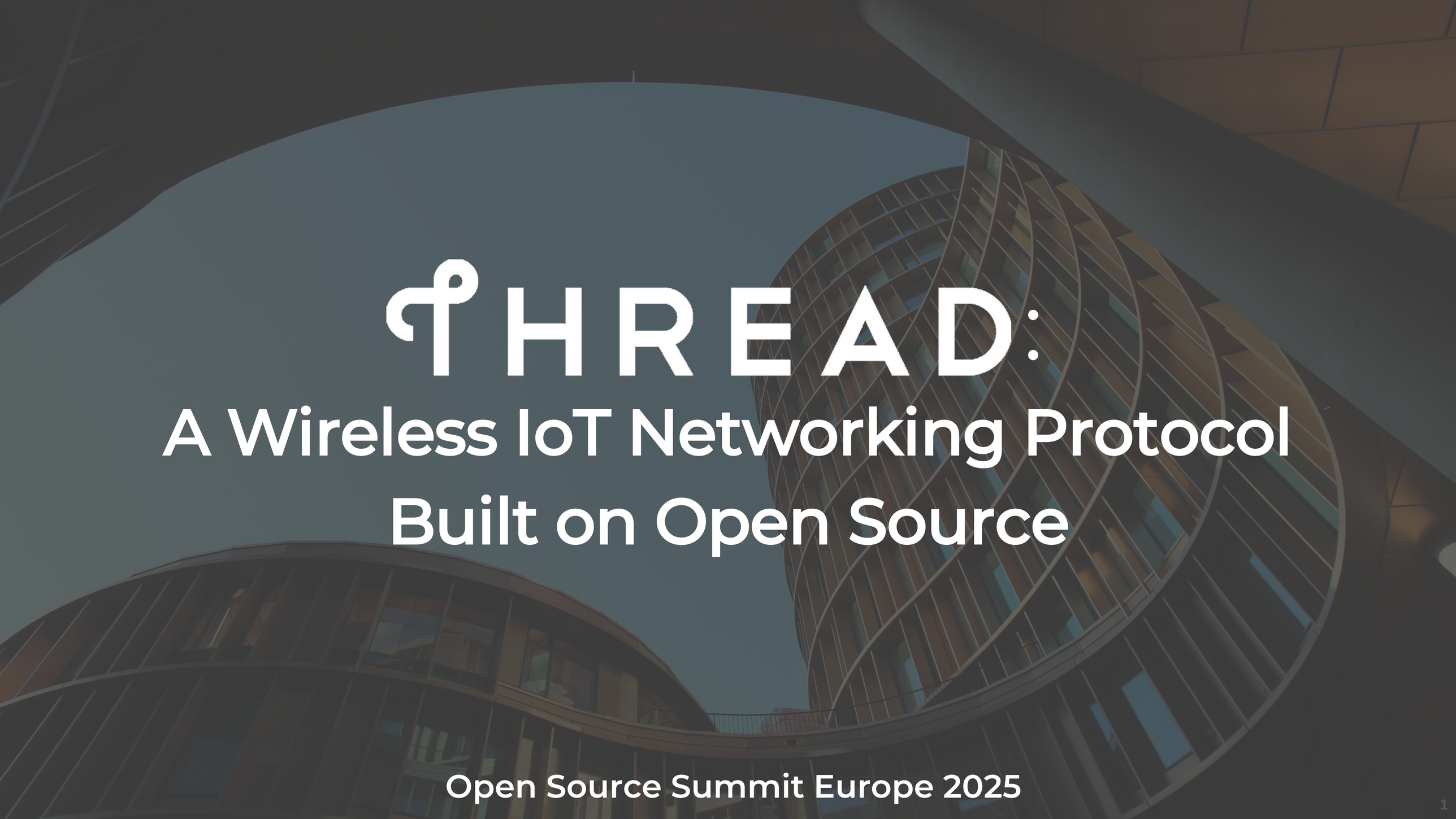


THREAD:
A Wireless IoT Networking Protocol Built on Open Source

Open Source Summit Europe 2025

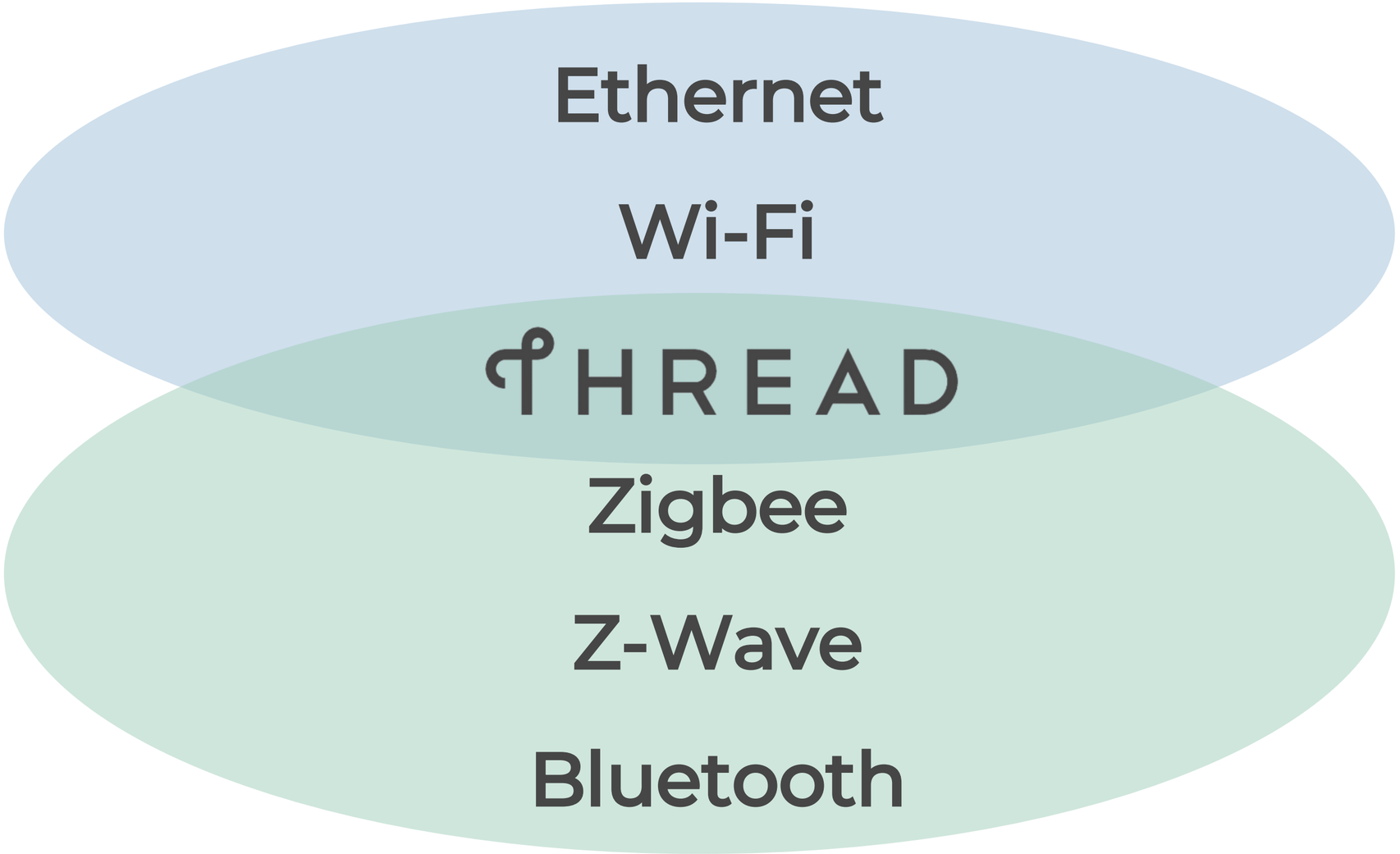
Speakers

Saurabh Kumar
Samsung Research America

Esko Dijk
IoTconsultancy.nl
Thread Group

EskoDijk

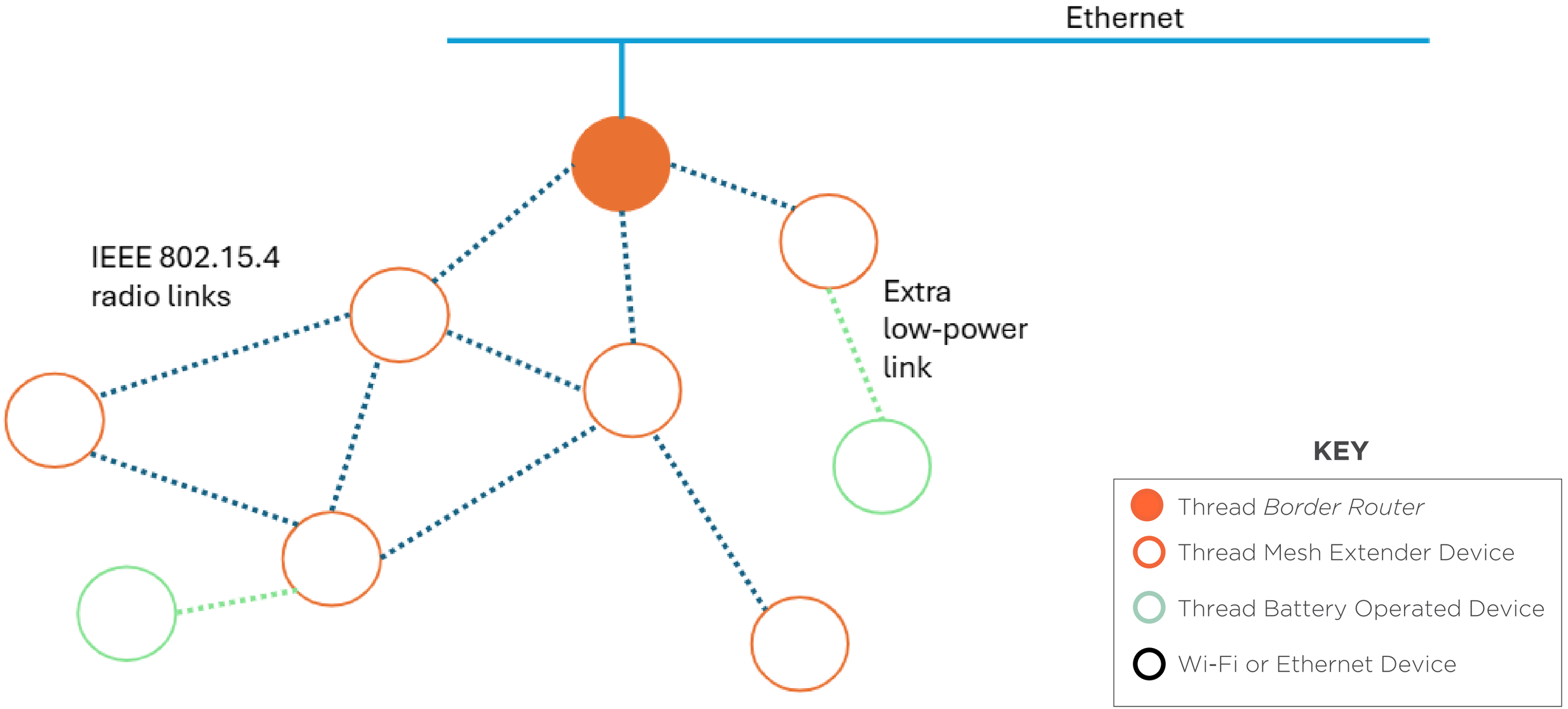
What is Thread?



Internet Protocol

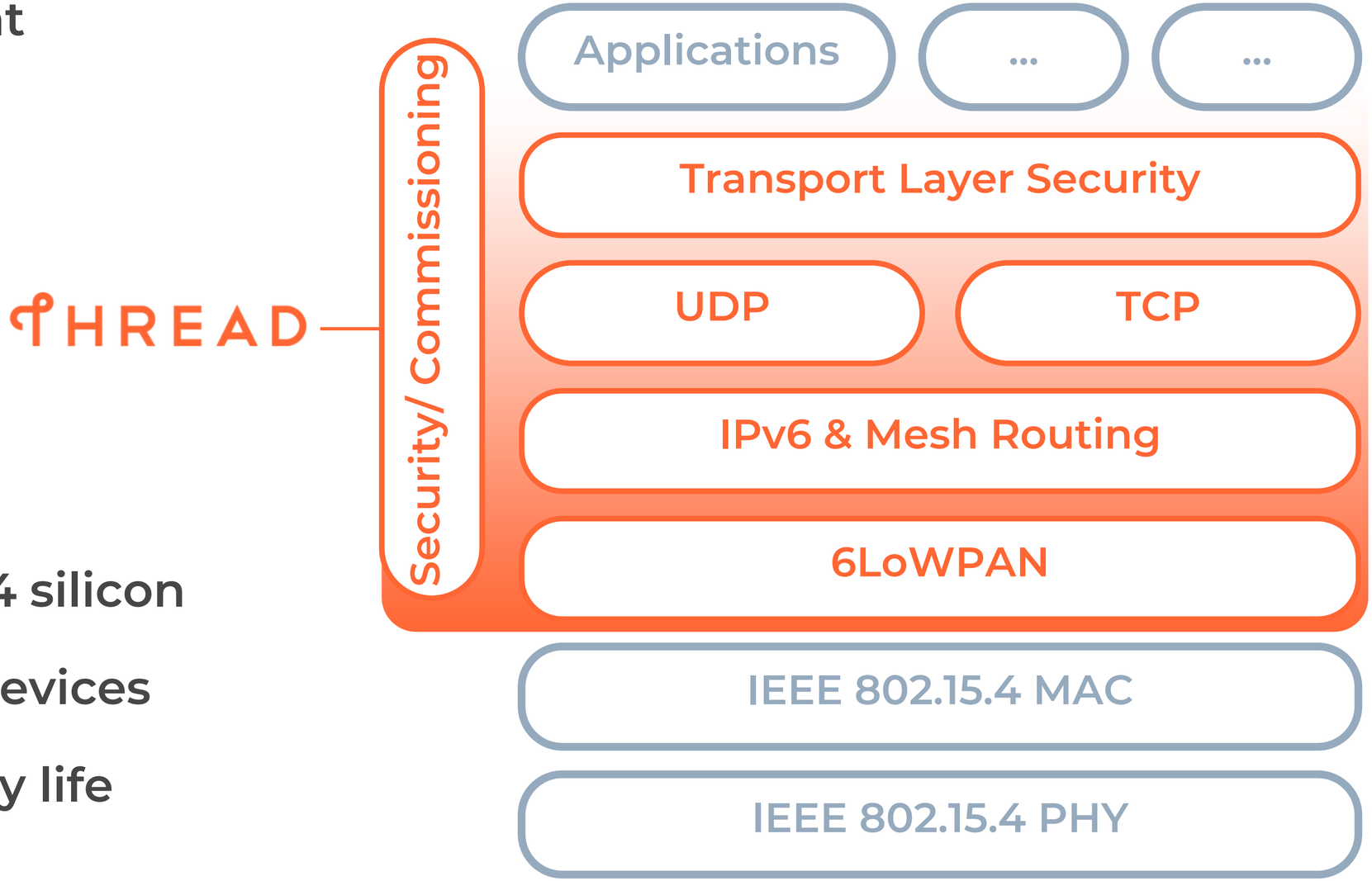
Low-power Wireless

Thread: Mesh Network



Thread: IP Stack for IoT Applications

- Use the technologies that power the internet



- Broadly available 802.15.4 silicon
- Tailored for embedded devices
- Low power & long battery life

About Thread Group

- **Delivering an impactful networking technology through a market-driven approach**
- **Educating the market on the benefits and uses of this technology**
- **Ensuring a great user experience through rigorous, meaningful product certification**
- **Certification via Test Lab**

THREADGROUP.ORG

Through industry collaboration, **Thread Group's mission** is to deliver an open and global wireless mesh networking protocol that extends IP infrastructure in homes and buildings. This low-power, reliable, and secure network enables the most diverse ecosystem of IoT devices.

Thread Specification: Release History

Thread 1.0 → 2015

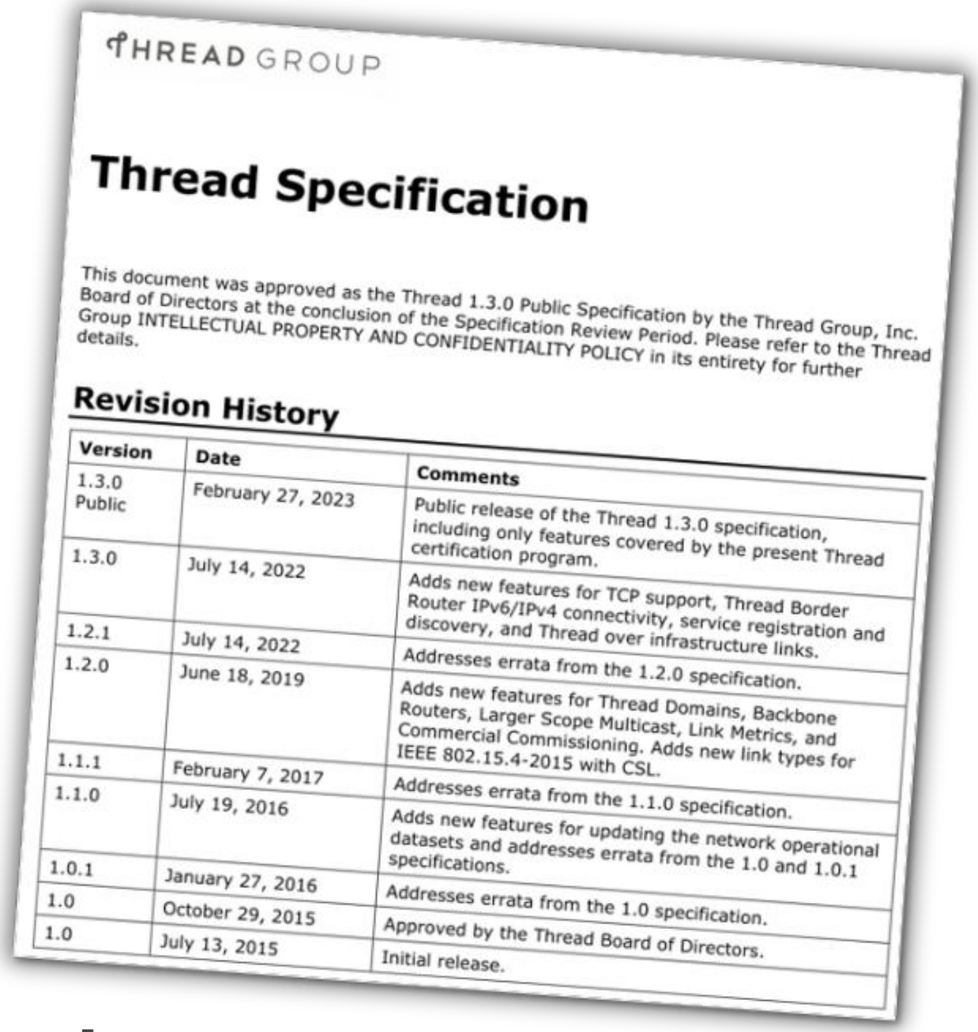
Thread 1.1 → 2016

Thread 1.2 → 2019

Thread 1.3 → 2022

Thread 1.4 → 2024

Thread x.y → Under Development



THREAD GROUP

Thread Specification

This document was approved as the Thread 1.3.0 Public Specification by the Thread Group, Inc. Board of Directors at the conclusion of the Specification Review Period. Please refer to the Thread Group INTELLECTUAL PROPERTY AND CONFIDENTIALITY POLICY in its entirety for further details.

Revision History

Version	Date	Comments
1.3.0 Public	February 27, 2023	Public release of the Thread 1.3.0 specification, including only features covered by the present Thread certification program.
1.3.0	July 14, 2022	Adds new features for TCP support, Thread Border Router IPv6/IPv4 connectivity, service registration and discovery, and Thread over infrastructure links.
1.2.1	July 14, 2022	Addresses errata from the 1.2.0 specification.
1.2.0	June 18, 2019	Adds new features for Thread Domains, Backbone Routers, Larger Scope Multicast, Link Metrics, and Commercial Commissioning. Adds new link types for IEEE 802.15.4-2015 with CSL.
1.1.1	February 7, 2017	Addresses errata from the 1.1.0 specification.
1.1.0	July 19, 2016	Adds new features for updating the network operational datasets and addresses errata from the 1.0 and 1.0.1 specifications.
1.0.1	January 27, 2016	Addresses errata from the 1.0 specification.
1.0	October 29, 2015	Approved by the Thread Board of Directors.
1.0	July 13, 2015	Initial release.

Smart Building & Home Automation

Smart Home

DALI

The world's leading technology for professional and industrial lighting, used everywhere from office buildings to industrial sites.

DALI+ uses Thread as its wireless carrier, delivering standardized, two-way dimming, diagnostics, and data exchange capabilities.

Thread brings wireless control into the DALI ecosystem, and when combined with DALI-2, enables hybrid systems that simplify integration in intelligent buildings.

Large scale DALI+ over Thread implementations are underway.

KNX®

One of the world's most ubiquitous building automation solutions in the world.
It's new IP-based KNX-IoT protocol expands the technology's benefits with IP network access.

KNX chose Thread as its wireless networking solution for its flexibility, scalability, and security.

At ISE 2025, demo showed Thread-enabled KNX IoT sensors, actuators and controllers
connected using Thread products with batteries lasting 5+ years.

KNX Startup Summit featured Thread.

matter

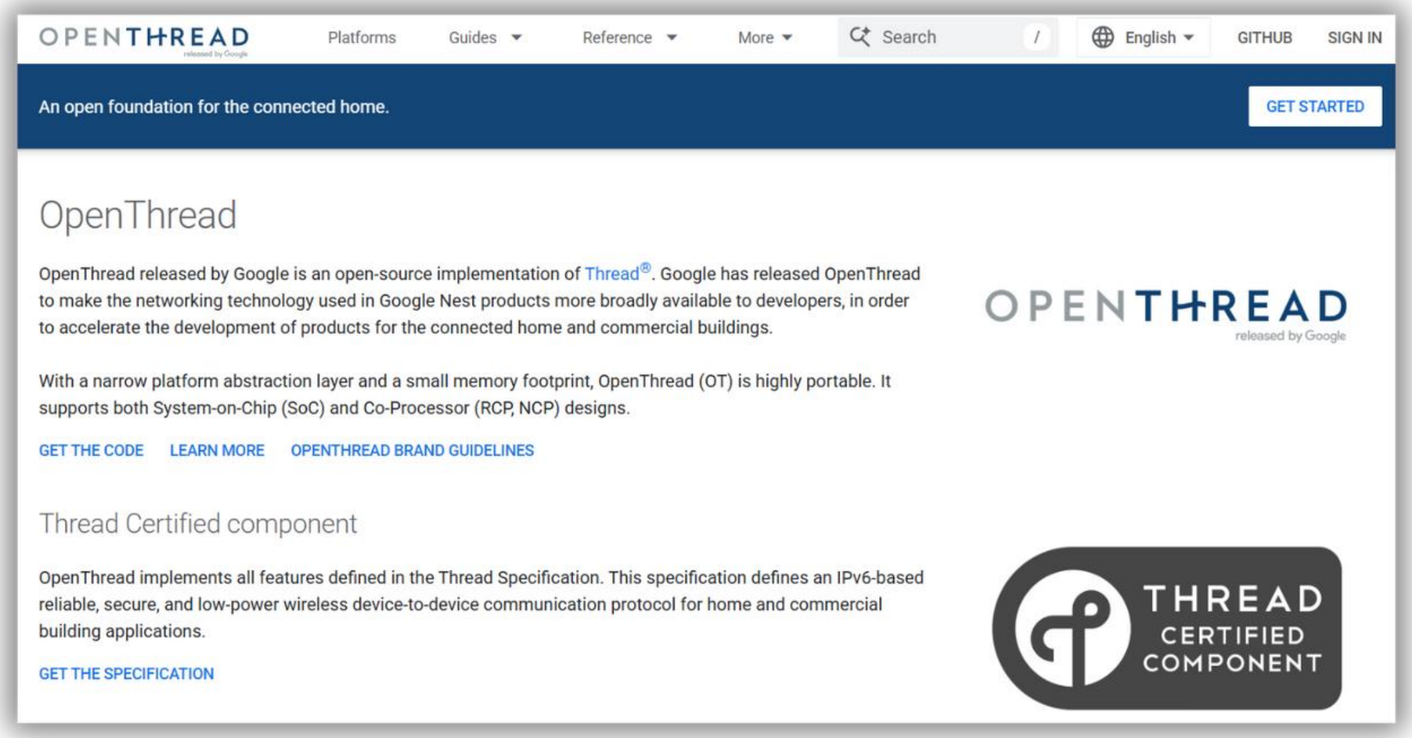
The application layer standard that lets developers and users connect their devices to the platforms of their choice.

Matter supports devices using Thread, as well as Ethernet, and Wi-Fi.

Device makers are building new kinds of devices and improving existing devices, while working seamlessly with the rest of the home.

OpenThread (OT)

- An open-source project, implementing the Thread standard
- The reference implementation for the standard
- Tools for Thread development & testing



openthread.io

Hosted on GitHub

OpenThread organization on GitHub

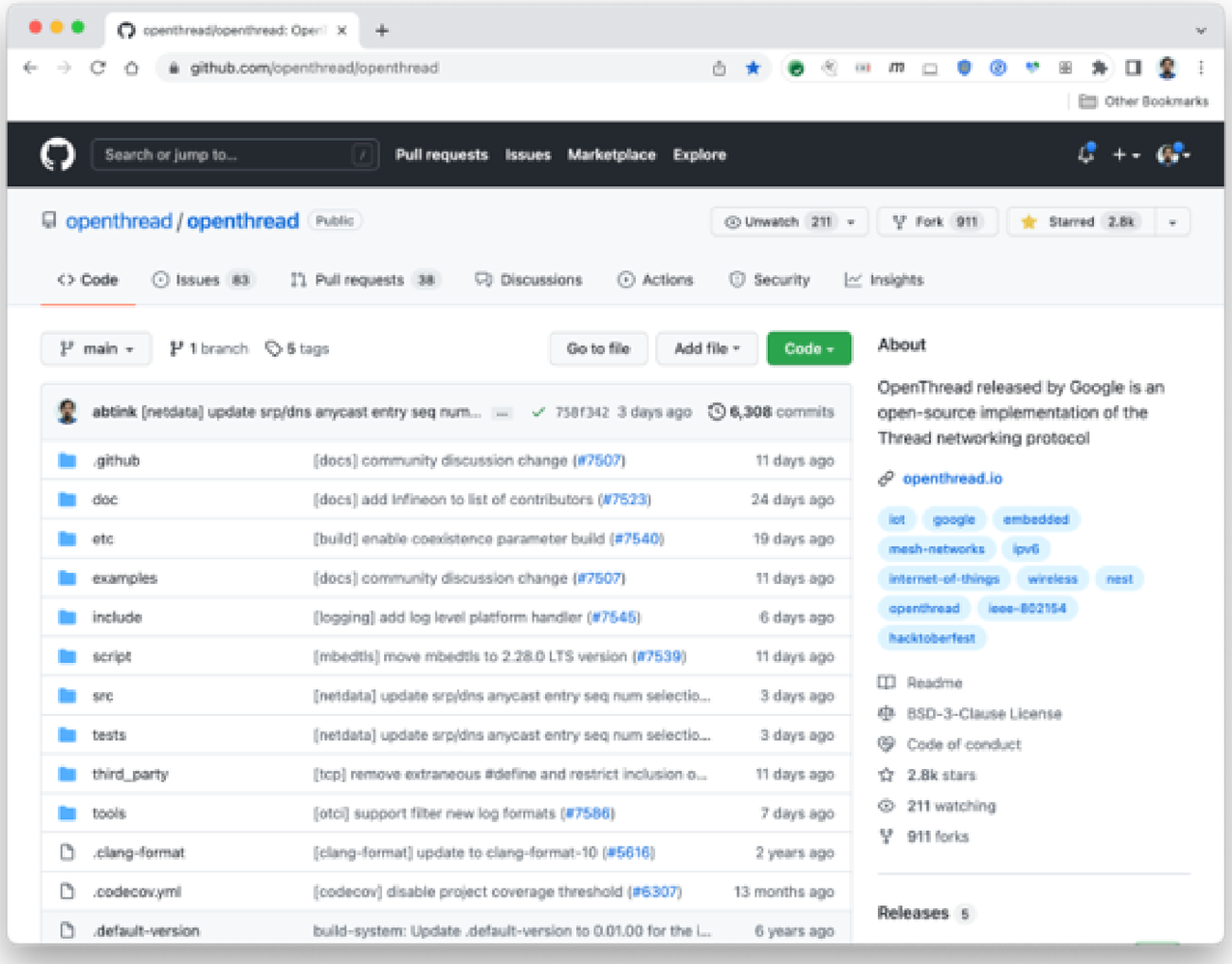
GitHub Enterprise funded by Google

GitHub Actions for continuous integration

GitHub Discussions for community engagement

Anyone can create and comment on:

- Issues
- Pull Requests
- Discussion Topics



License: BSD-3-Clause

Copyright (c) 2022, The OpenThread Authors

Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

1. Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.

2. Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.

3. Neither the name of the copyright holder nor the names of its contributors may be used to endorse or promote products derived from this software without specific prior written permission.

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

Silicon Vendor Support

arm

CASCODA
Innovation in IoT

ESPRESSIF

infineon

NORDIC SEMICONDUCTOR

NXP

qorvo

QUALCOMM
QUALCOMM TECHNOLOGIES, INC.

SAMSUNG

SILICON LABS

STMicroelectronics

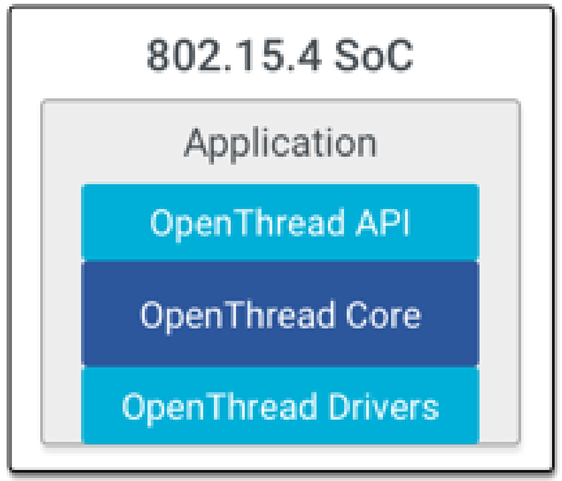
SYNOPSYS®

Telink

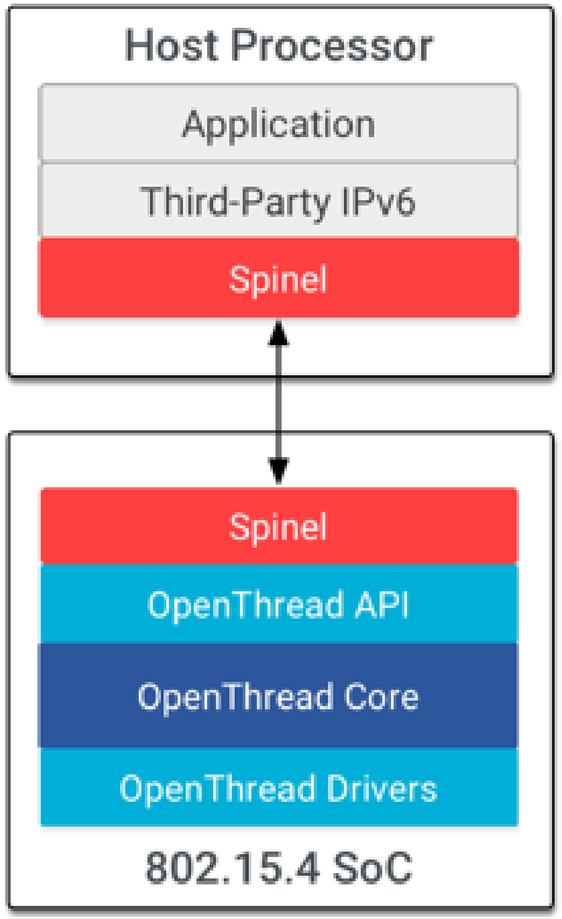
Texas Instruments

OpenThread: Architecture Options

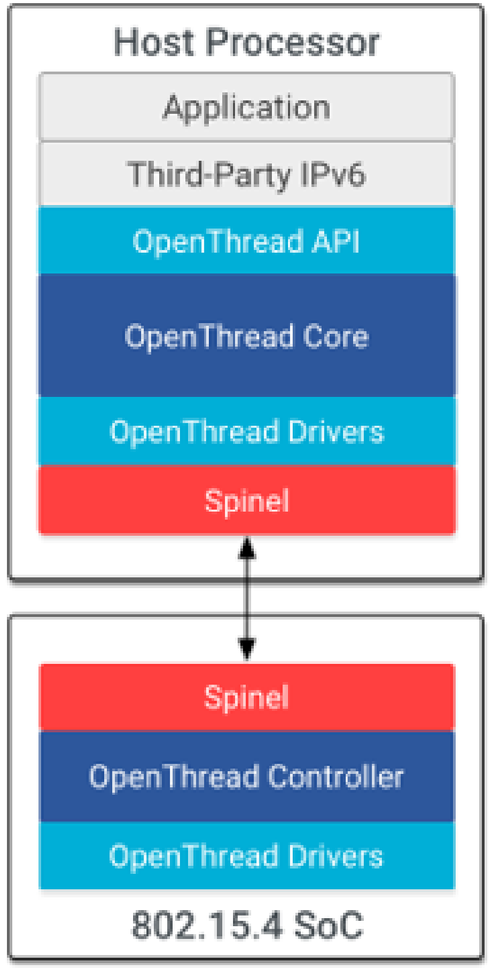
Single Chip



Multiple-Chip Network Co-Processor (NCP)



Multiple-Chip Radio Co-Processor (RCP)

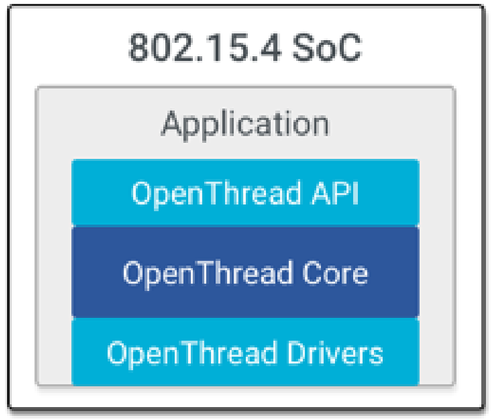


OpenThread (OT) APIs & Core

OT Public APIs

ot{Module}{FunctionName} - **C99** functions, types, and constants

Ex: otThreadSetEnabled(), otDatasetGetActive(), otSrpClientGetServices()



OT Core

C++11

Does not use standard library (STL)

OT Platform APIs

otPlat{Module}{FunctionName} - **C99** functions, callbacks, types and constants

Device drivers, OS related: alarm, radio, non-volatile settings, UART/SPI

Ex: otPlatRadioTransmit(), otPlatAlarmMilliStartAt(), otPlatSettingsSet(), OT_ERROR_NONE

OpenThread Modules & Layered Architecture

Common	Timer, TrickleTimer, Tasklet, Uptime, Settings Tlv, Data, Array, Message, LinkedList, Pool, String, Error, Encoding, BitVector Log, Notifier, Random
Mac	Radio, SubMac, Links, Mac, DataPollHandler, DataPollSender, LinkRaw MeshForwarder, IndirectSender, CslTxScheduler, MessageQueue, PriorityQueue
Network	Lowpan, Ip6, Mpl, Udp, Icmp6, Tcp, Socket Coap, CoapSecure, Netif
Thread	Mle, MleRouter, Tmf, DiscoverScanner, AddressResolver, KeyManager, NetDiag, Mlr ChildTable, RouterTable, NeighborTable NetworkData:: Local, Leader, Publisher, Service, Notifier
MeshCoP	Dataset, ActiveDataset, PendingDataset, DatasetManager, DatasetUpdater Joiner, JoinerRouter, Commissioner, BorderAgent
BorderRouter	RoutingManager
Utils	Srp::Client, Srp::Server, Dns::Client, Dns::Resolver, Dns::Dso HistoryTracker, PingSender, ChildSupervision, JamDetector, ChannelMonitor

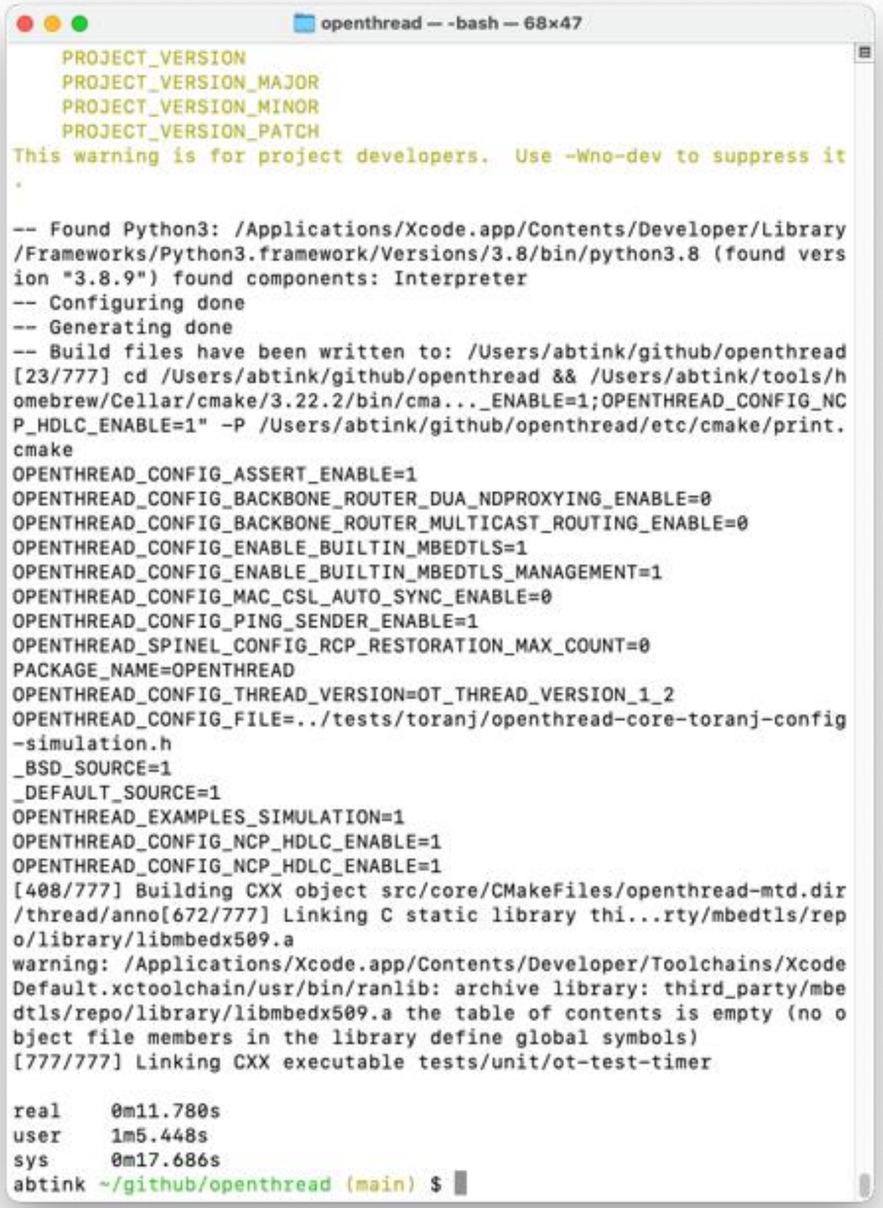
OpenThread Build System & Configs

Build system (makefiles)

CMake (and ninja)
BUILD.gn

Three high level configurations:

OPENTHREAD_FTD - build Full Thread Device
OPENTHREAD_MTD - build Minimal Thread Device
OPENTHREAD_RADIO - build RCP firmware



OpenThread Configs

Many configuration options:

openthread/src/core/config/*.h

OPENTHREAD_CONFIG_{FEATURE}_ENABLE

To enable/disable feature

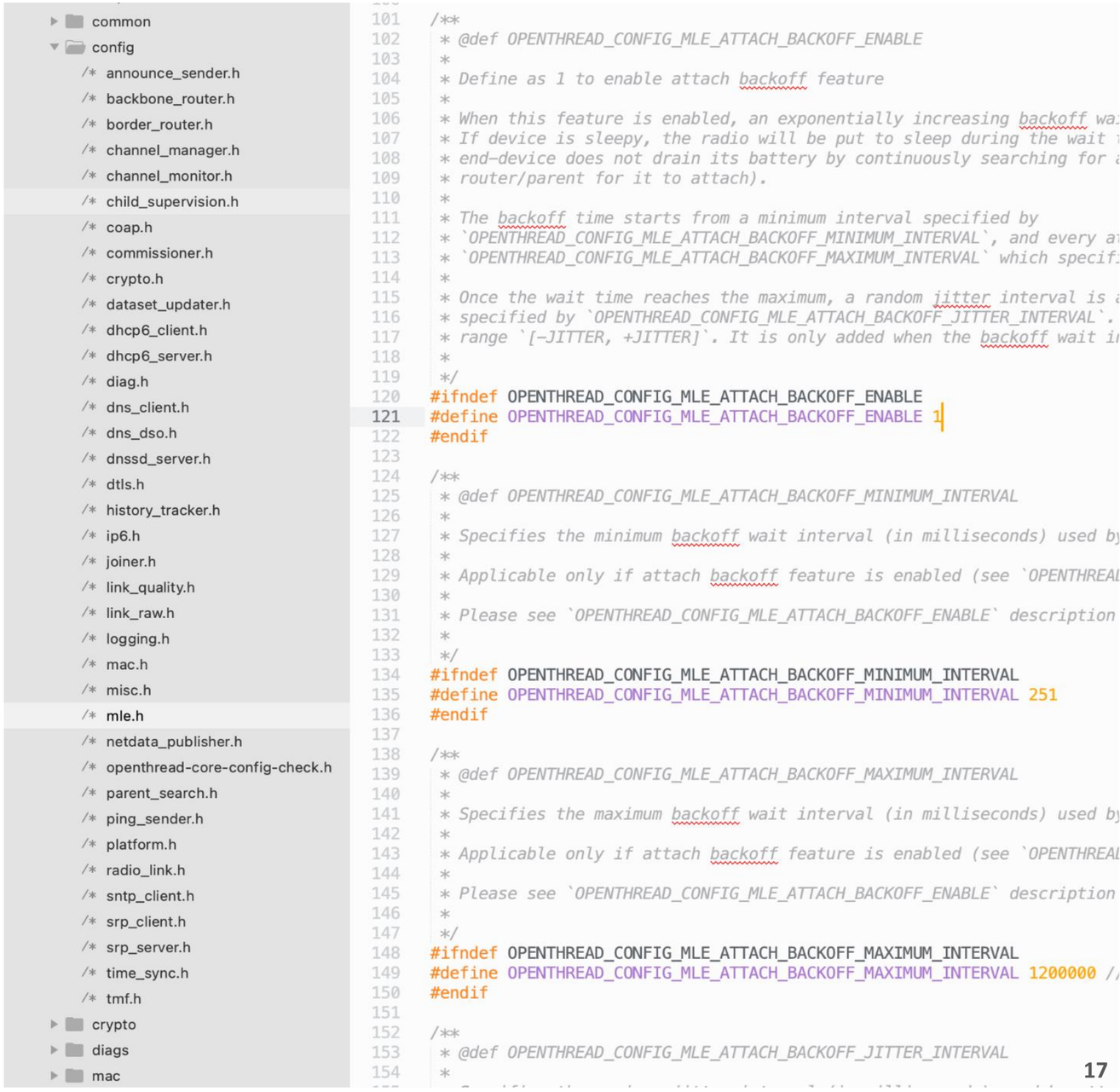
OPENTRHEAD_CONFIG_{FEATURE}_{PARAM}

Parameter related to a feature

OT stack will sanity check the configs

Error if removed/renamed config is used

Error on conflicting configs



OpenThread Logging

Log Level

Debg, Info, Note, Warn, Crit

Set statically by a build config

Feature to set dynamically at run-time

Log Output

Sent to platform `otPlatLog` or app (CLI/NCP) or disabled

Log Configurations

Log line is configurable, whether to include:

Uptime

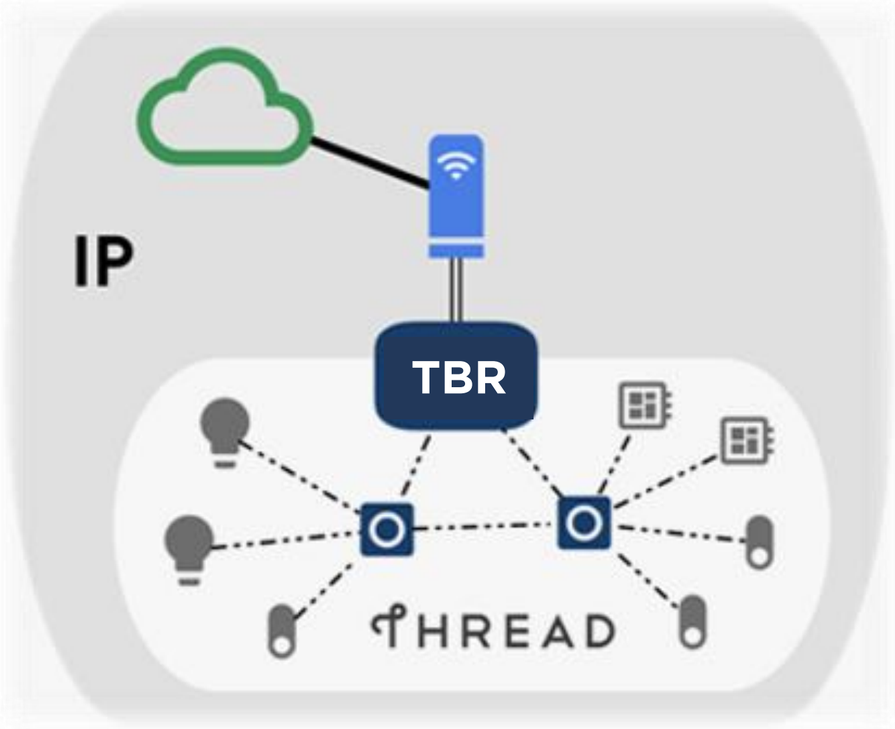
Log level

Any suffixes

```
00:00:06.894 [D] SubMac--------: RadioShortAddress: 0xdc00
00:00:06.894 [N] Mle-----------: Role detached -> leader
00:00:06.894 [N] Mle-----------: Partition ID 0x658b9609
00:00:06.894 [I] RouterTable---: Route table
00:00:06.894 [I] RouterTable---:     55 0xdc00 - me - leader
00:00:06.894 [I] Notifier------: StateChanged (0x100012a5) [Ip6+ Role Rloc+ Partit
00:00:06.894 [I] Settings------: Saved NetworkInfo {rloc:0xdc00, extaddr:564e8ca88
00:00:06.894 [I] Settings------: ... pid:0x658b9609, mlecntr:0x3ef, maccntr:0x3e8,
00:00:06.894 [D] Mle-----------: Store Network Information
00:00:06.894 [I] Mle-----------: Send Data Response (ff02:0:0:0:0:0:0:1)
00:00:06.894 [I] Settings------: Re-saved NetworkInfo {rloc:0xdc00, extaddr:564e8c
00:00:06.894 [I] Settings------: ... pid:0x658b9609, mlecntr:0x3ef, maccntr:0x3e8,
00:00:06.894 [D] Mle-----------: Store Network Information
00:00:06.894 [I] BbrLeader-----: PBBR state: None
00:00:06.894 [D] MlrManager----: MlrManager::UpdateReregistrationDelay: rereg=0, r
00:00:06.894 [I] AnnounceSender: Started
00:00:06.894 [D] MlrManager----: -------- Multicast Addresses --------
00:00:06.894 [D] Mac-----------: Request to start operation "TransmitDataDirect"
00:00:06.894 [D] Mac-----------: Starting operation "TransmitDataDirect"
00:00:06.894 [D] SubMac--------: RadioState: Receive -> CsmaBackoff
00:00:06.894 [D] SubMac--------: RadioState: CsmaBackoff -> Transmit
00:00:06.897 [D] SubMac--------: RadioState: Transmit -> Receive
00:00:06.897 [D] Mac-----------: ==============================[TX len=075]=======
00:00:06.897 [D] Mac-----------: | 41 D8 4D CE FA FF FF 54 | 59 EC 81 A8 8C 4E 56
00:00:06.897 [D] Mac-----------: | 3B 01 F0 4D 4C 4D 4C E7 | B9 00 15 06 00 00 00
```

Thread Border Router (TBR)

- A device role defined in the Thread specification
- Interfaces high-speed Ethernet/Wi-Fi to low-speed Thread mesh network
- Performs traffic limiting, IP forwarding, NAT64 (IPv6/IPv4 translation), 6LoWPAN compression & decompression, …
- Application-agnostic (not an ‘app gateway’!)
- Various commercial products are certified Thread Border Routers (name some products?)

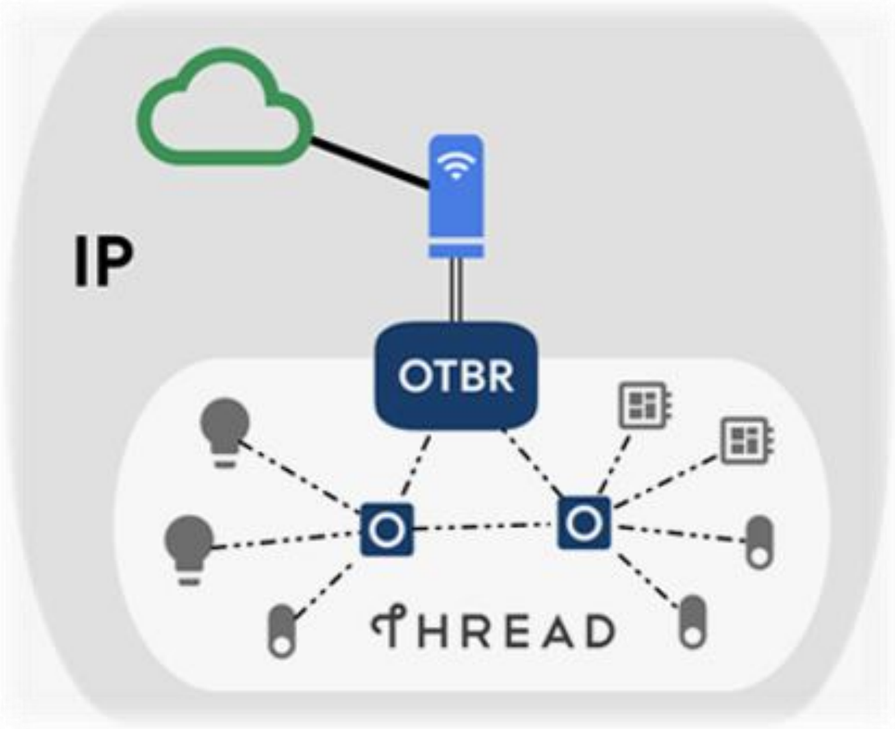


OpenThread Border Router (OTBR)

- Separate Github repos (link, link) – includes OpenThread repo by reference
- Typically built on Raspberry Pi with an 802.15.4 USB dongle
- OTBR guide on openthread.io

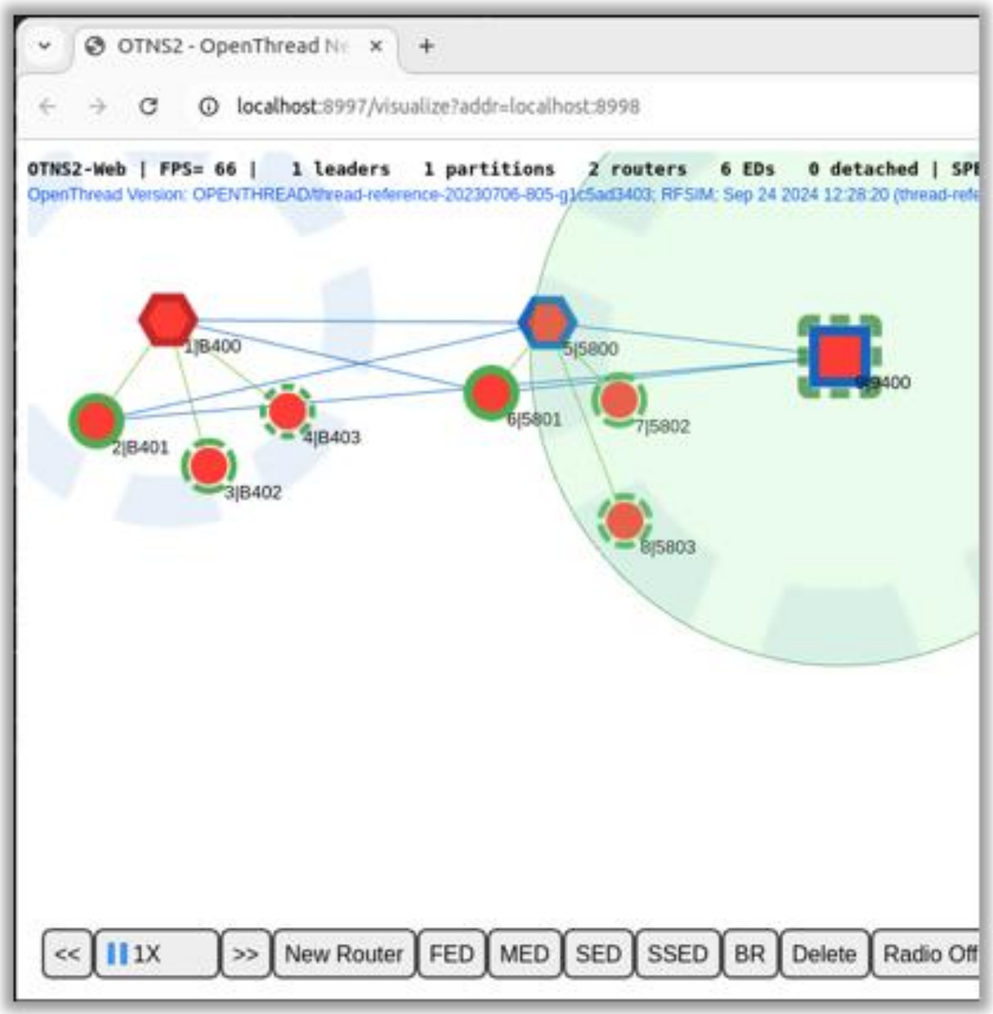
- Easiest method is to build the OTBR reference release – by script

- Custom TBR build: allows more experimenting & tweaking than commercial TBRs
  - Full CLI access



OpenThread Network Simulator (OTNS)

- Quickly test OpenThread nodes in any mesh network configuration!
- Run faster than real-time (e.g. 100x)
- Runs the actual OpenThread stack code; if needed multiple firmware/Thread versions

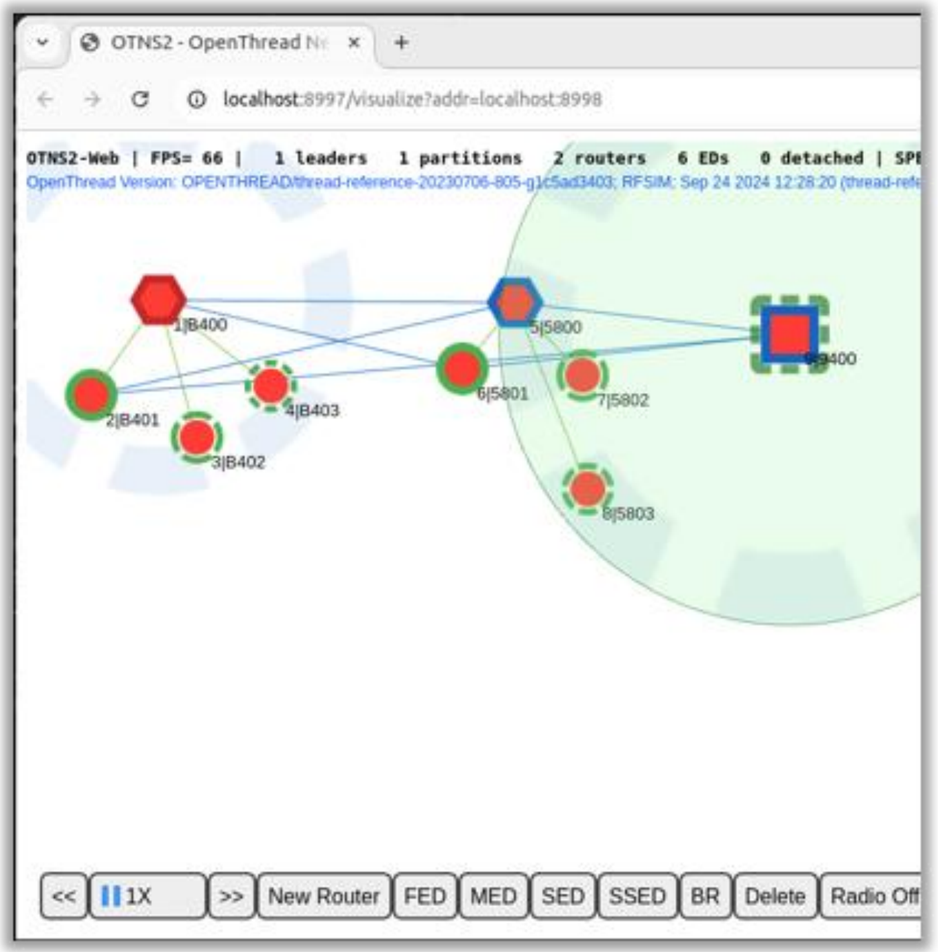


OpenThread Network Simulator (OTNS)

- Accurately models radio chip timing per node with CCA, back-off, interference, …
- Simulates radio propagation(indoor/outdoor) & RSSI with statistical models

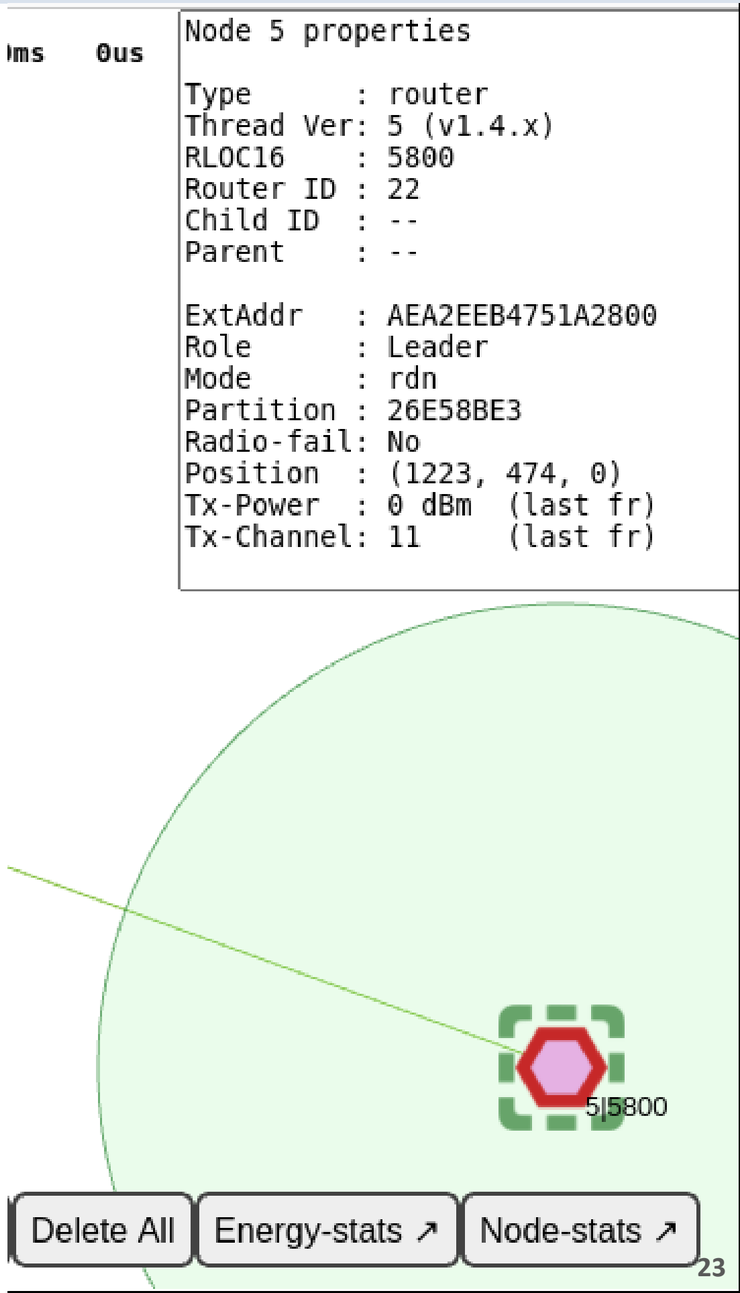
- Repo: https://github.com/openthread/ot-ns

- Languages used:
  - Go (simulator)
  - C/C++ (OpenThread nodes, sim-platform)
  - Python 3 (scripting & examples)
  - Bash script (build & install)



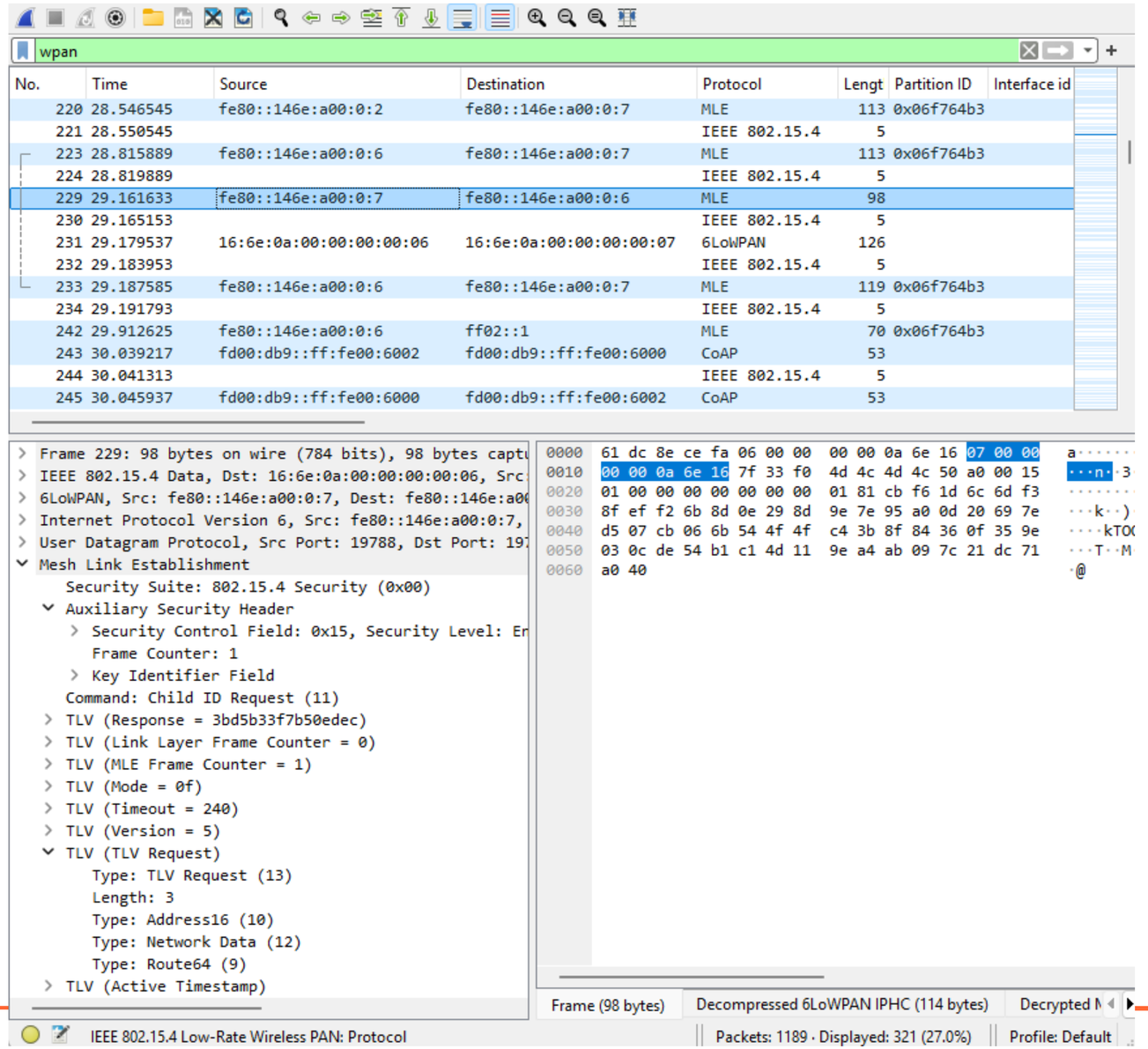
Integrating OTNS Simulation into the Dev Cycle

- Simulated nodes: much faster to deploy & test than actual hardware
- Corner cases can be explicitly created - by setup script
- Bugs or unexpected side effects are faithfully reproduced
- Modify OpenThread code, add features, and see the result – in node CLI output, in Wireshark captures, or in node logs



IPv6 & Traffic Analysis

- IPv6 on the Thread mesh network
- Enables protocol dissection using standard Wireshark
- Works the same for real (captured) radio traffic, as well as OTNS simulated radio traffic



Getting Started with OTNS

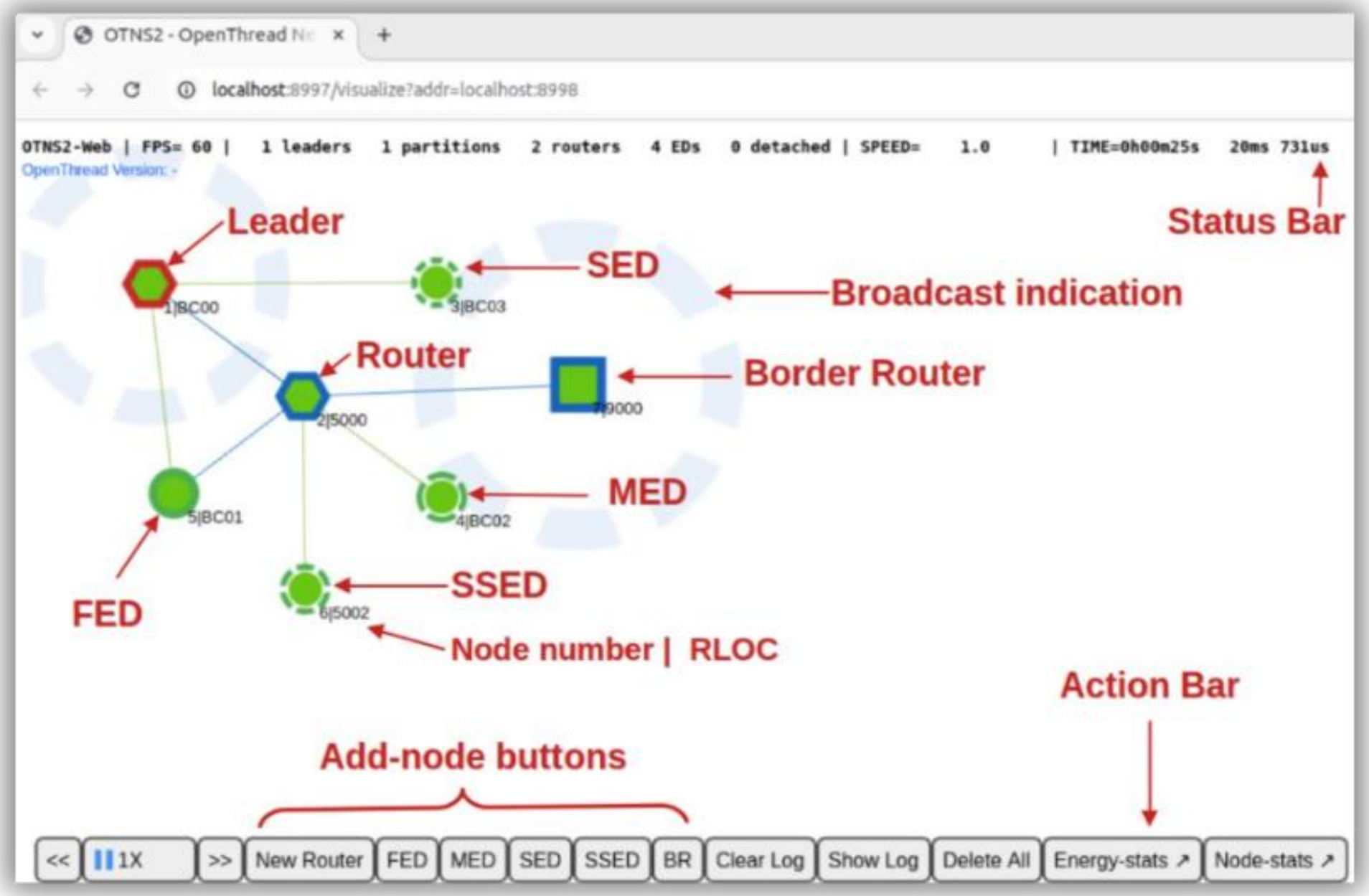
- MacOS or Linux – scripted install
- Docker “playground” for trying out:

```
docker run -it -p 8997-9000:8997-9000 openthread/otns2:playground
```

- Start using and understanding Thread mesh networks – without any hardware investment!
- Any OpenThread/Thread learnings are directly applicable to real hardware nodes & networks

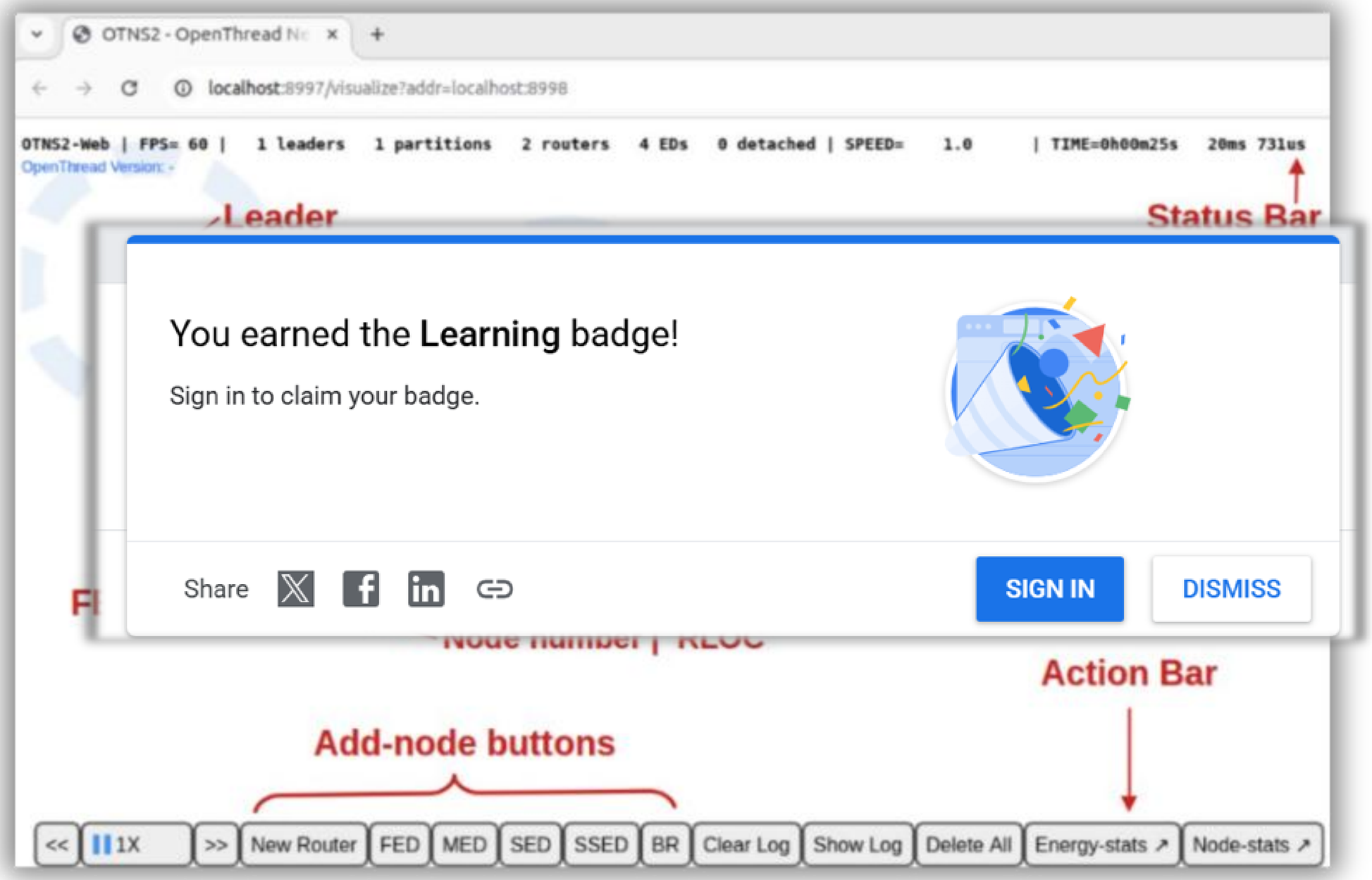
Getting Started

1-hour tutorial: [Codelab]
on openthread.io



Getting Started

1-hour tutorial: [Codelab]
on openthread.io



Your code can build the future of IoT !
openthread.io
github.com/openthread
THREAD
Open Source Summit Europe 2025