OpenThread Training

17.12.2020 – ÁRPÁD NAGY

Agenda

- Thread
- OpenThread
- OpenThread within GeckoSDK
- Commissioning
- OT – Border Router
- Q&A

Thread

Background and Concepts

Background

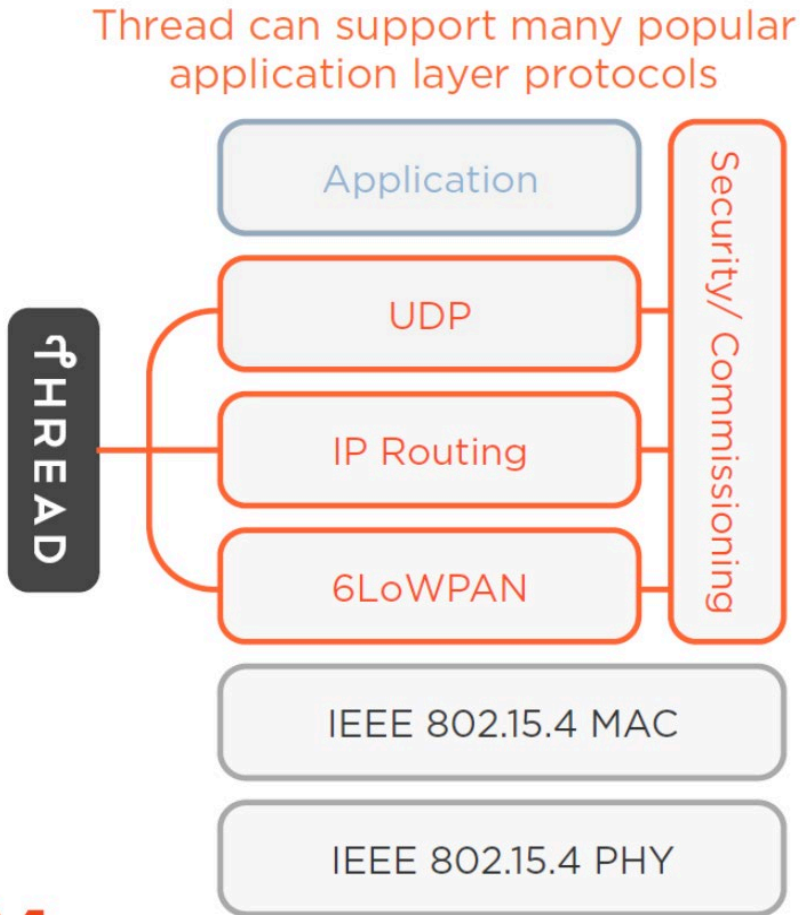
Products to communicate with **each other**, **cloud services** and the **customer**.

- Requirements:
  - Secure
  - Scalable
  - Resilient
  - Low Power
  - IP-Based

Overview

Build on Existing Technologies

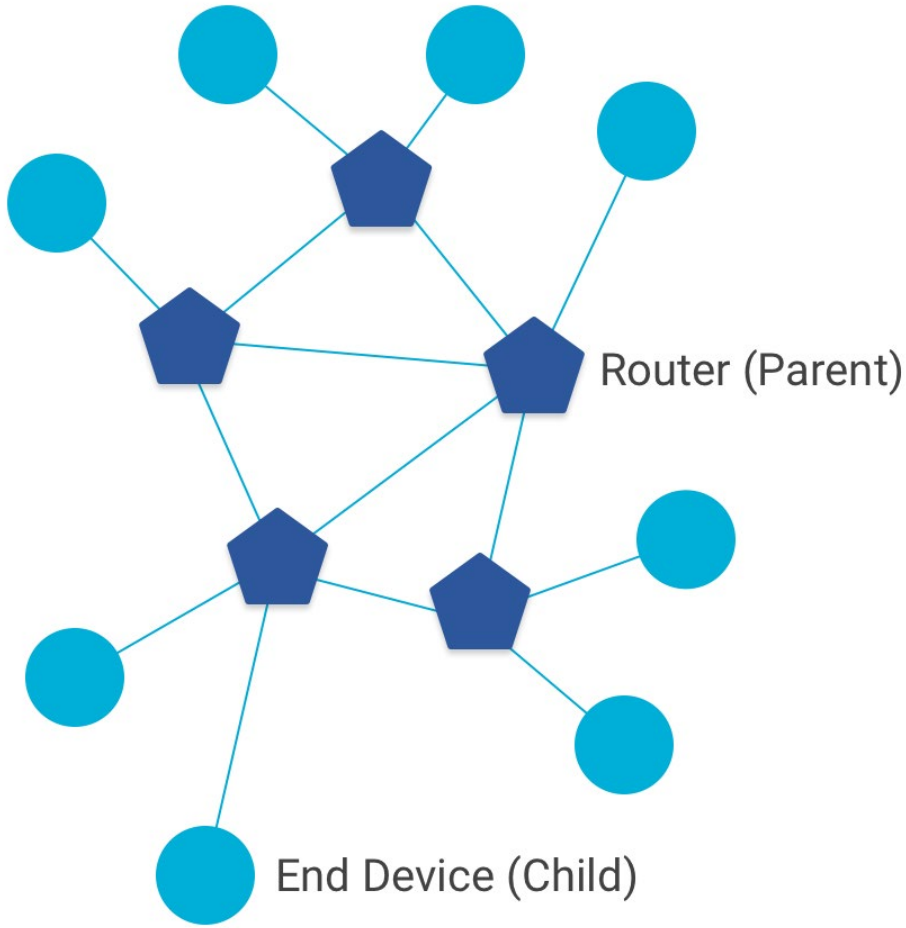
- Same PHY as Zigbee (802.15.4)
  - Fast time to market
- IETF Link layer standards (6LoWPAN)
- Security / Simplicity
- Efficiency
- Thread Specification (1.1)



Network Overview

Scalable Mesh Network

- Up to **32 routers** per network
- Up to **511 end devices** per network
- **Parent-Child** relationship



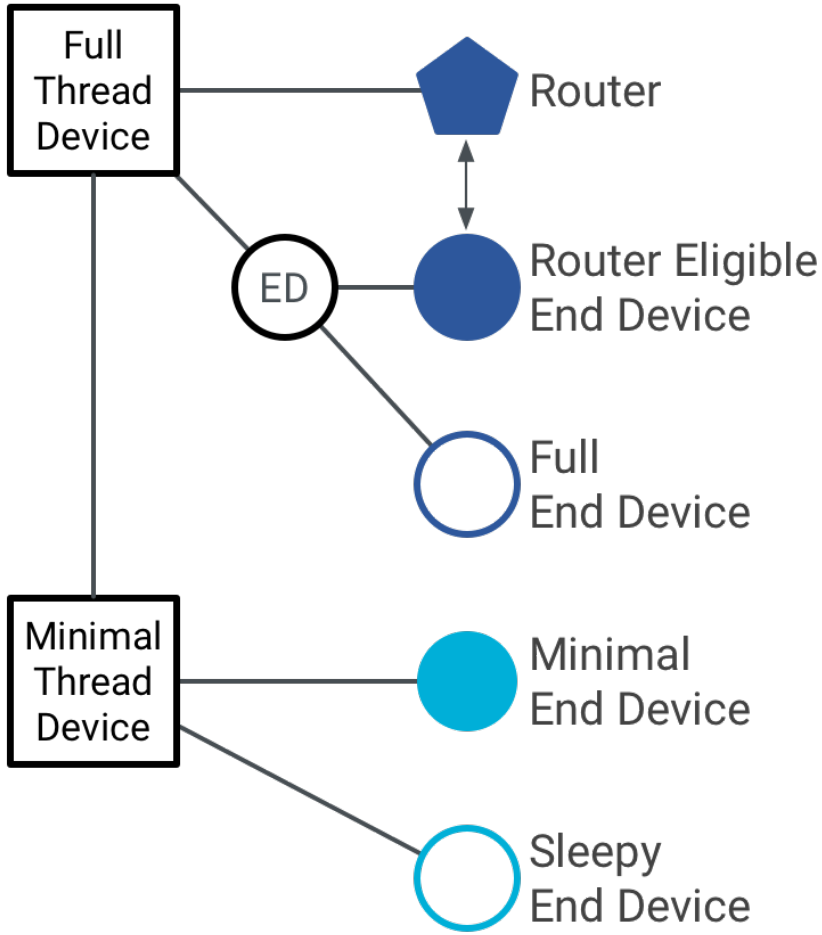
Network Overview

Full Thread Device

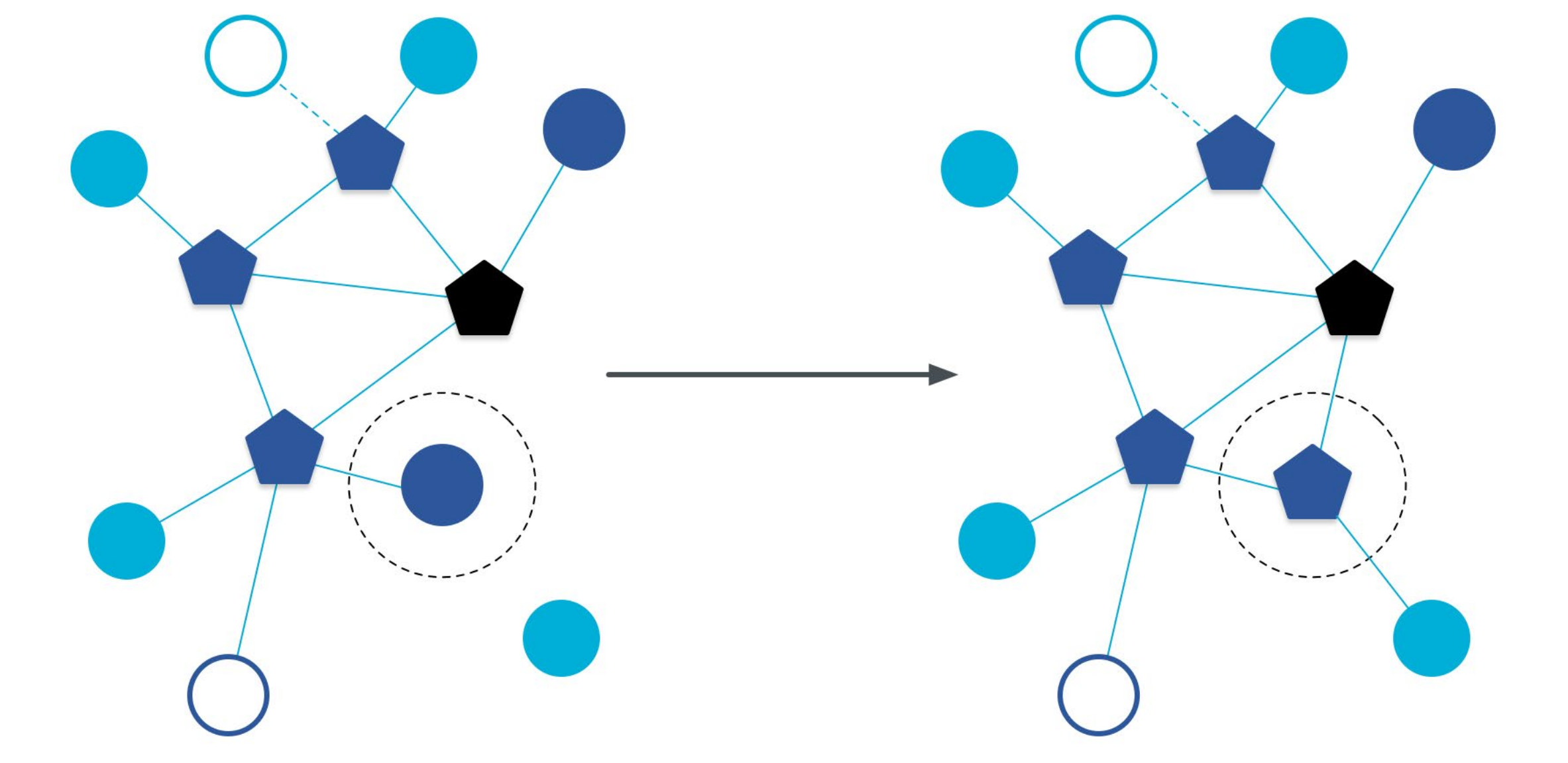
- Radio on at all times
- Router multicast address
- 3 main types: **Router, REED, FED**

Minimal Thread Device

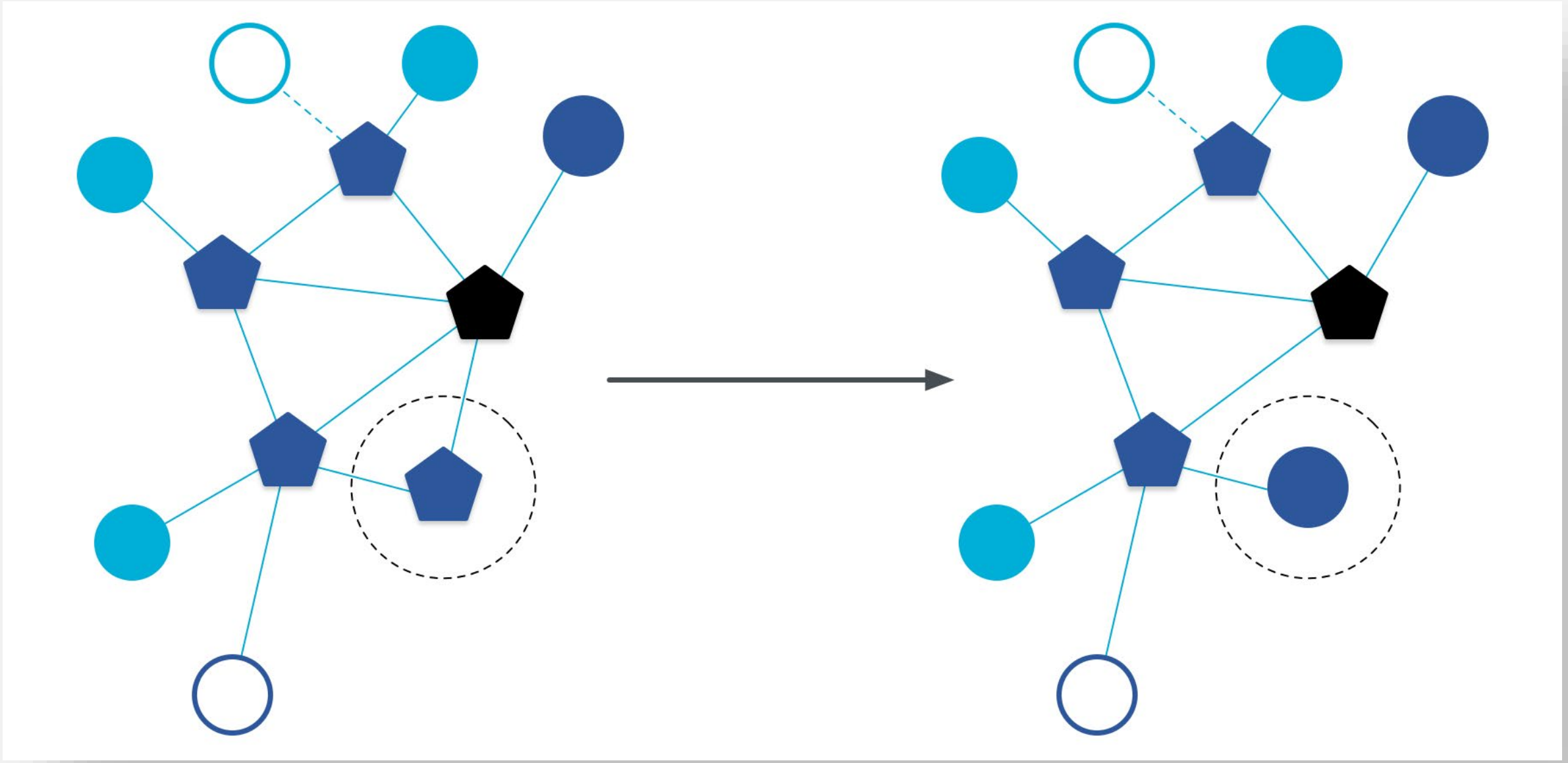
- All messages to the parent
- No Router multicast address
- 2 main types: **MED, SED**



Self scaling - Upgrade



Self scaling – Downgrade



Other device roles

Border Router

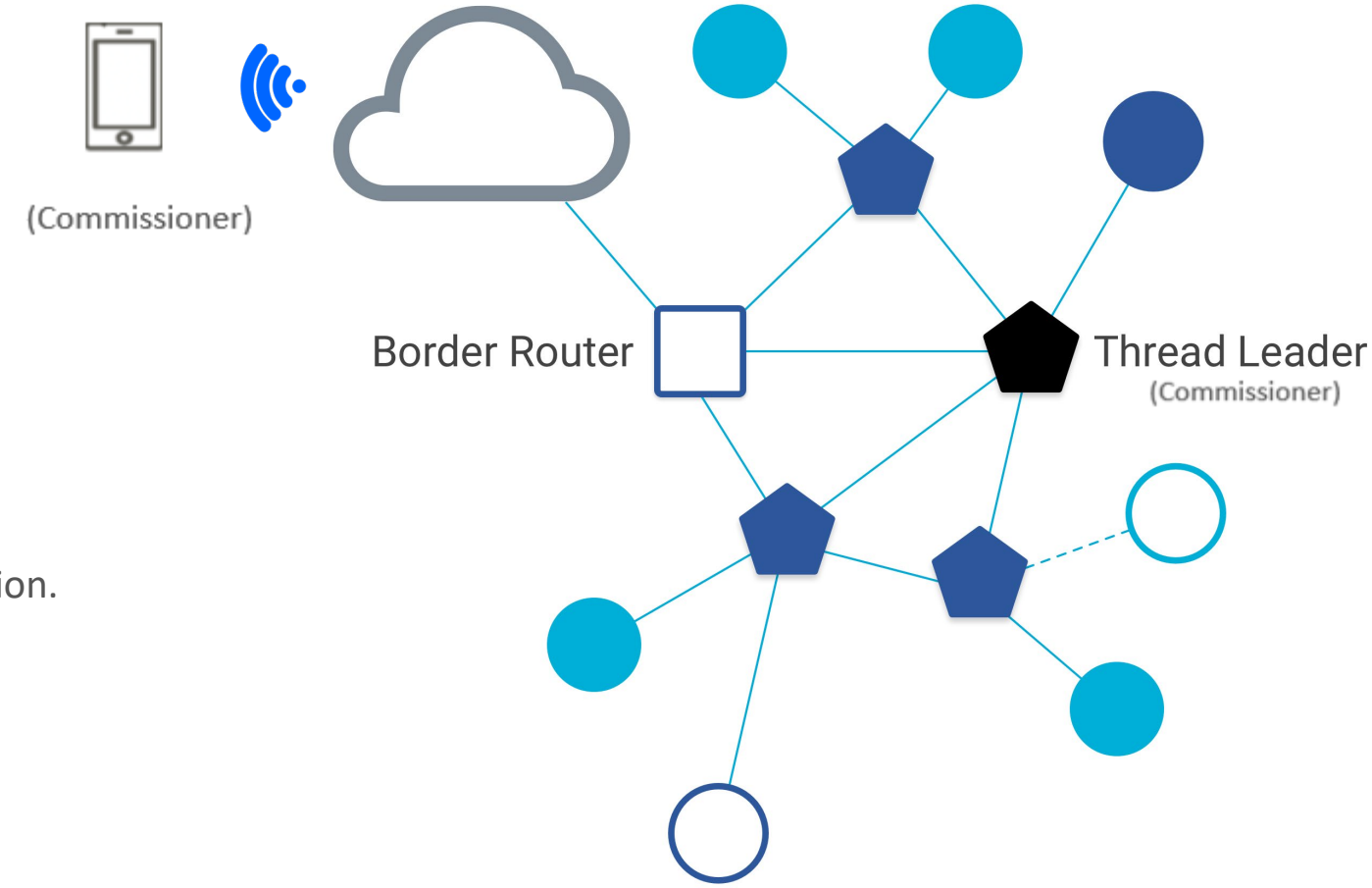
- Bridge between Thread – nonThread
- Configure external connectivity

Thread Leader

- Manage routers.
- Self elected dynamically.
- Aggregates and distributes network configuration.

Commissioner

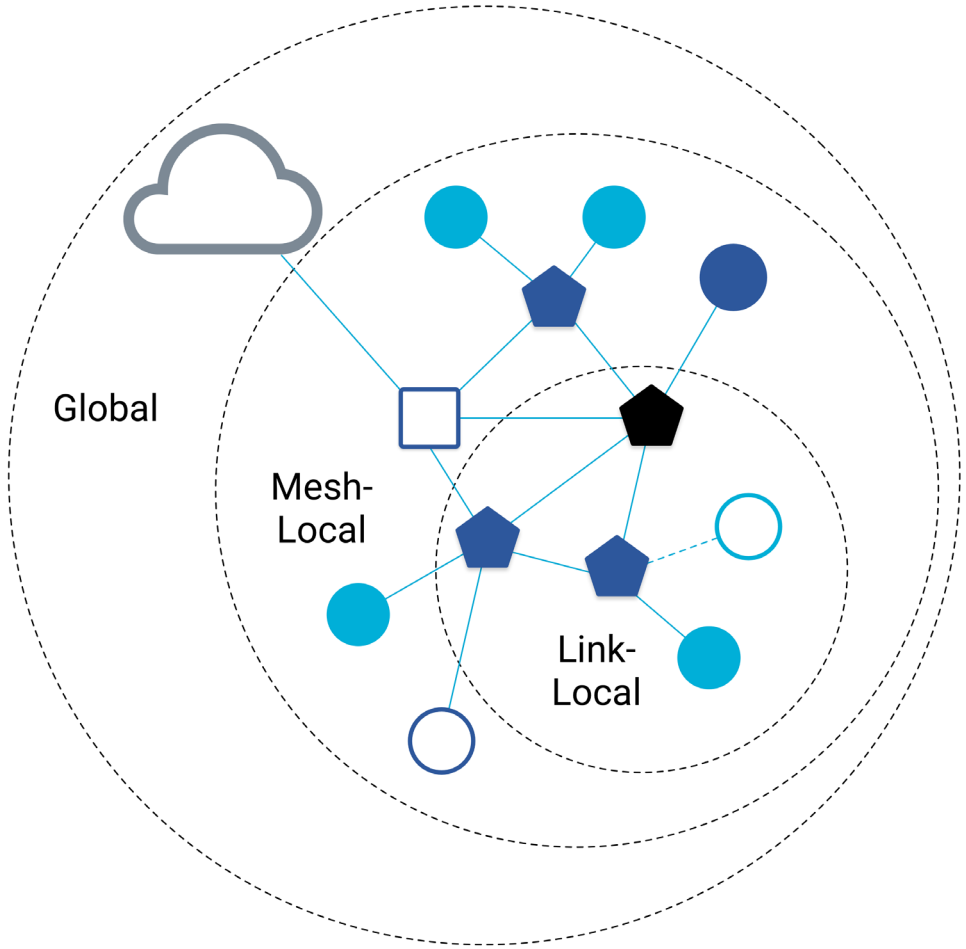
- Authenticates joining devices. More later.



Addressing Scopes

- **IPv6 – ramp it up!**
  - fe80:0000:0000:0000:0202:b3ff:fe1e:8329
  - fe80:0:0:0:202:b3ff:fe1e:8329
  - fe80::202:b3ff:fe1e:8329

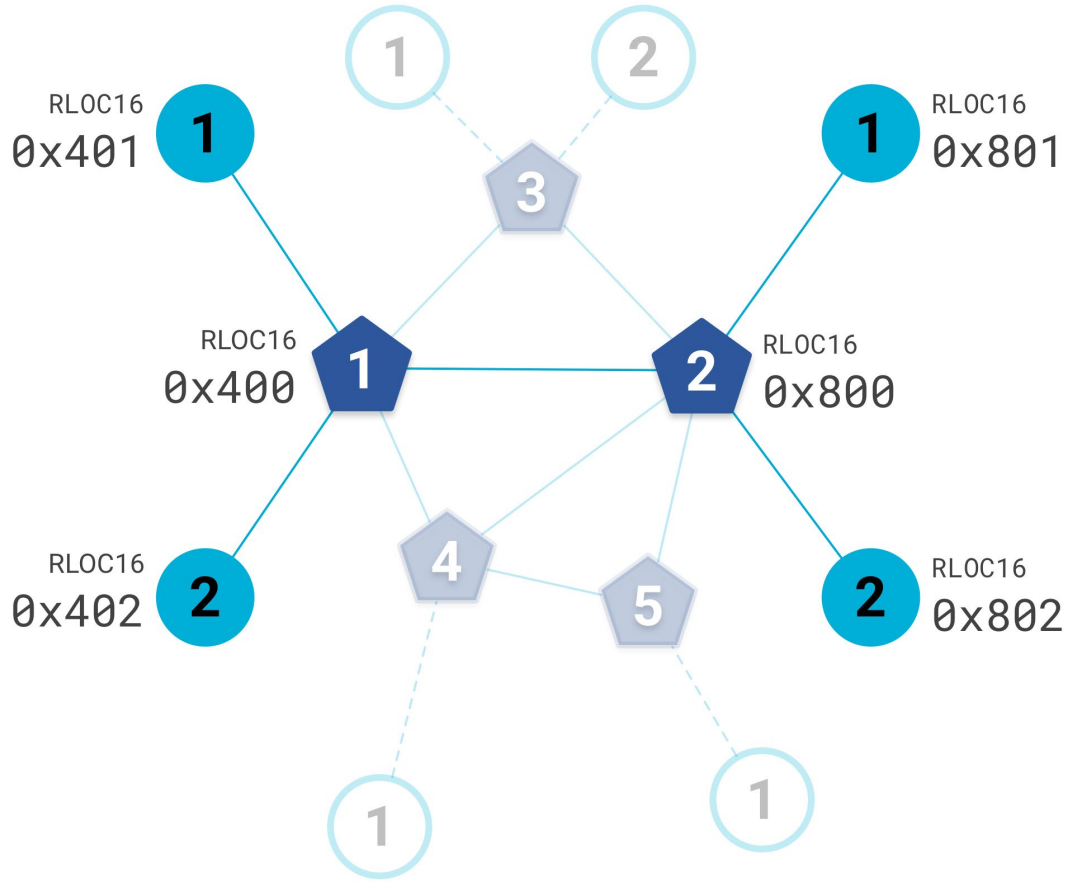
- Link-Local – Single transmission range
  - Prefix fe80::/16 -> fe80:: - fe80:ffff:….
- Mesh-local – Addresses in the network
  - Prefix fd00::/8 -> fd00:: - fdff:fff….
- Global – All reachable addresses



RLOC – Routing Locator

- Router ID + Child ID
  - Unique /device in mesh (Zigbee Short ID)
  - Change based on location

- Interface Identifier (IID)
  - 0000:00ff:fe00:RLOC16 – ff:fe00:401
- RLOC = MeshLocal + IID
  - Ex: MeshLocal Prefix = fd01:4b1d:ee40:1::/64
  - Ex: IID : 0000:00ff:fe00:0401
  - **RLOC: fd01:4b1d:ee40:1::ff:fe00:0401 IPv6 address**



Unicast address types

	Desciption	Example	IID	Scope
LLA	Identifier for single hop.	fe80::54db:881c:3845:57f4	Based on 802.15.4 Extended Address	Link-Local
ML-EID / ULA	Topology independent unique	fde5:8dba:82e1:1:416:993c:8399:35ab **fd00::/8 prefix**	Random after Commissioning	Mesh-Local
RLOC	Identify based on location.	fde5:8dba:82e1:1::ff:fe00:1001	0000:00ff:fe00:***RLOC16***	Mesh-Local
ALOC	Identify via RLOC lookup	fde5:8dba:82e1:1::ff:fe00:fc01	0000:00ff:fe00:fc***XX***	Mesh-Local
GUA	Global identifier outside Thread	2000::54db:881c:3845:57f4 – prefix 2000::/3	Based on IP provisioning	Global

Network Formation and Discovery

- Unique Identifiers – PAN ,XPAN , Network NAME
  - 0xBEEF, 0xBEEFCAFEDEADFFFF, silabsThreadNetwork
- 3 common steps of forming / joining
  1. 802.15.4 active scan
  2. Beacon Broadcast
  3. Repeat on each channel

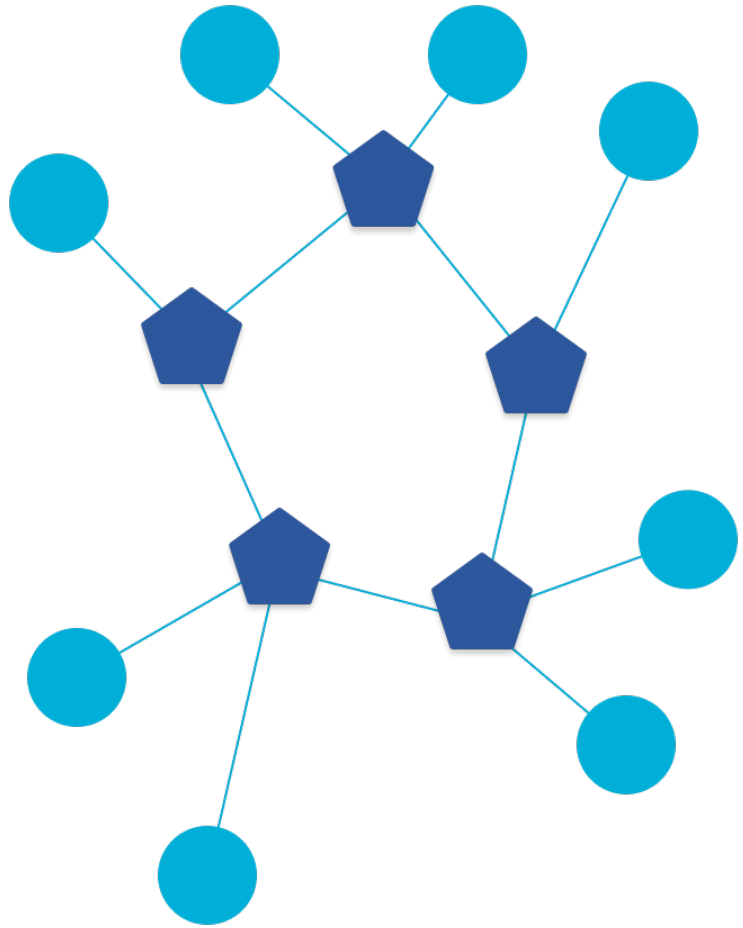
Creating a network

1. Select least busy channel
2. Select unused PAN
3. Become Router
4. Elect yourself as leader.

- Joining a network
  - Configure network info from Beacon
  - MLE Attach process
  - Join as End device

Router selection

- Form a **ConnectedDomainSet, Router only path**
- Distributed algorithm maintains CDS
- Add routers to:
  - Increase **Path diversity**
  - Maintain **redundancy**
  - Extend **connectivity**
  - Router **threshold 16**
- Remove routers to:
  - Reduce routers below **max 32.**
  - Allow routers elsewhere.

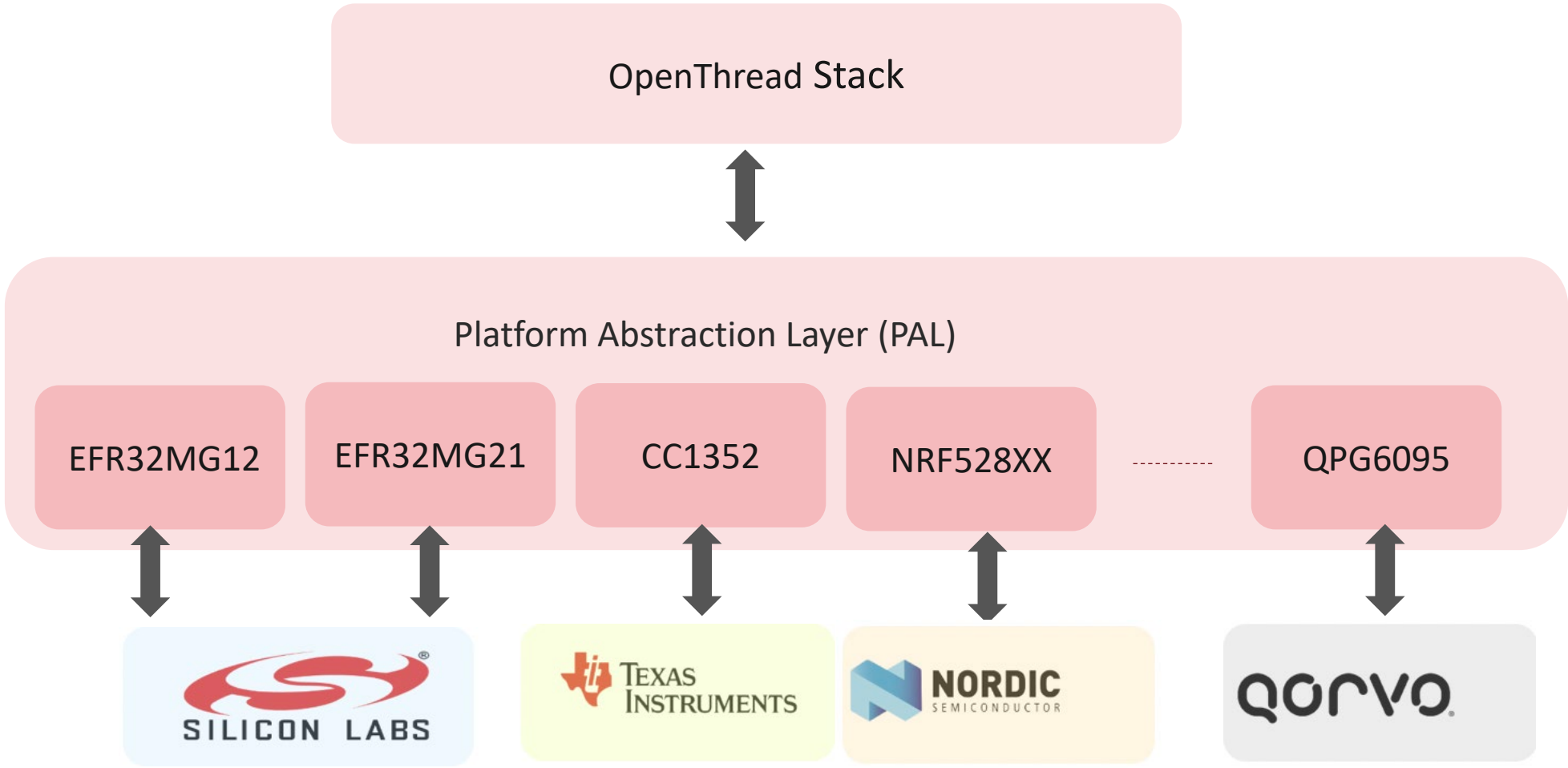


OpenThread - https://openthread.io/

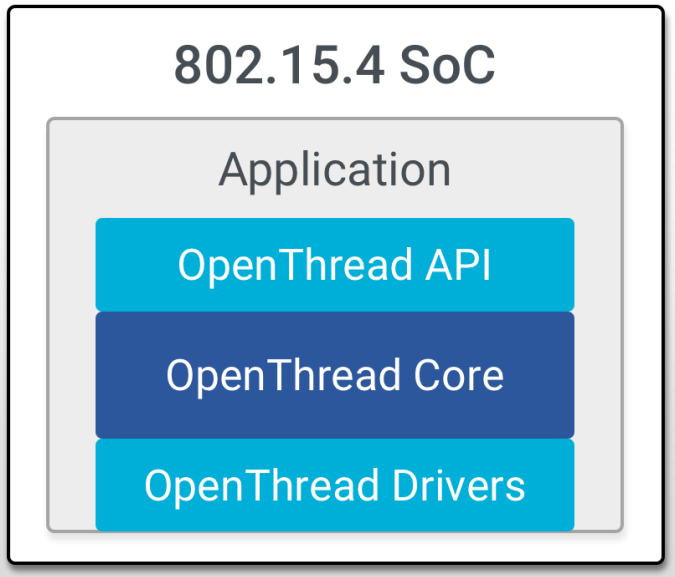
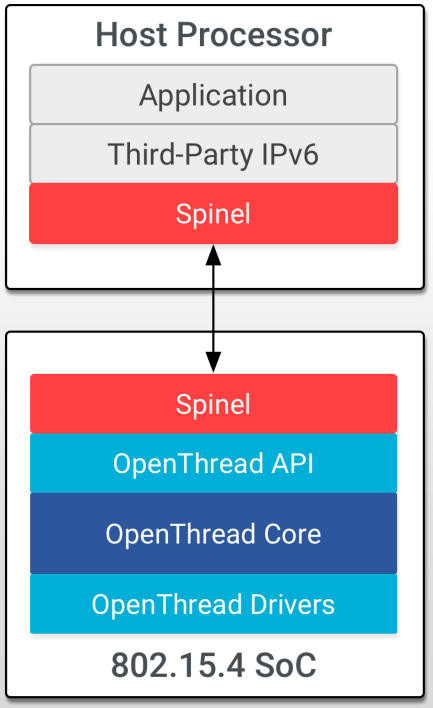
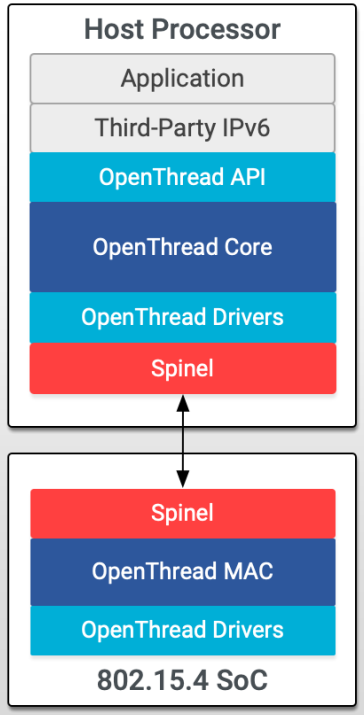
OpenThread

- Open Source, C++ implementation of Thread
- OS and Platform agnostic
- Thread Certified
- Supported on multiple platforms from different vendors

What's inside?



Architecture

SOC	NCP	RCP
		
Everything on one processor. Usually End Devices. Other variant: Multi – interface.	Thread runs on SoC. App layer on Host that can Sleep. Usually Gateways.	Thread and App layer on Host. Low level MAC layer on SoC. Thread can utilize more resources.

How it works?

- Get a POSIX platform (MAC, Linux, Raspberry Pi, VM or Docker)
- Clone the repository
  - git clone –recursive https://github.com/openthread/openthread.git
- Setup the environment
  - cd openthread
  - ./bootstrap
- APIs in /include/openthread
  - API Reference information openthread.io/reference.
- Samples in /examples/apps
  - Make –f examples/Makefile-efr32mg12
- Binaries generated to /output/<platform>/bin

```
pi@raspberrypi:~/git/openthread $ ls -lh --group-directories-fir
total 1.3M
drwxr-xr-x  2 pi pi 4.0K Nov 24 20:39 autom4te.cache
drwxr-xr-x  4 pi pi 4.0K Nov 24 20:40 doc
drwxr-xr-x  5 pi pi 4.0K Nov 24 20:14 etc
drwxr-xr-x  4 pi pi 4.0K Nov 24 20:40 examples
drwxr-xr-x  3 pi pi 4.0K Nov 24 20:40 include
drwxr-xr-x  2 pi pi 4.0K Nov 24 20:14 script
drwxr-xr-x  7 pi pi 4.0K Nov 24 20:40 src
drwxr-xr-x  6 pi pi 4.0K Nov 24 20:40 tests
drwxr-xr-x 13 pi pi 4.0K Nov 24 20:40 third_party
drwxr-xr-x  7 pi pi 4.0K Nov 24 20:40 tools
-rw-r--r--  1 pi pi 347K Nov 24 20:39 aclocal.m4
-rw-r--r--  1 pi pi  27K Nov 24 20:14 Android.mk
-rw-r--r--  1 pi pi  639 Nov 24 20:14 AUTHORS
-rwxr-xr-x  1 pi pi 2.5K Nov 24 20:14 bootstrap
-rw-r--r--  1 pi pi 3.2K Nov 24 20:14 BUILD.gn
-rw-r--r--  1 pi pi 7.5K Nov 24 20:14 CMakeLists.txt
-rw-r--r--  1 pi pi 3.2K Nov 24 20:14 CODE_OF_CONDUCT.md
-rwxr-xr-x  1 pi pi 730K Nov 24 20:40 configure
-rw-r--r--  1 pi pi  36K Nov 24 20:14 configure.ac
-rw-r--r--  1 pi pi 5.9K Nov 24 20:14 CONTRIBUTING.md
-rw-r--r--  1 pi pi 1.5K Nov 24 20:14 LICENSE
-rw-r--r--  1 pi pi 6.8K Nov 24 20:14 Makefile.am
-rw-r--r--  1 pi pi  37K Nov 24 20:40 Makefile.in
-rw-r--r--  1 pi pi 1.3K Nov 24 20:14 NOTICE
-rw-r--r--  1 pi pi 6.8K Nov 24 20:14 README.md
-rw-r--r--  1 pi pi  14K Nov 24 20:14 STYLE_GUIDE.md
```

Configuration

- Compile-time constants /src/core/config
- Makefile build switches /examples/common-switches.mk
- Building samples with switches
  - Make –f examples/Makefile-efr32mg12 COMMISSIONER=1 JOINER=1
- Platform specific Build options
  - /examples/platforms/efr32/src/openthread-core-efr32-config.h
- Determine which sample is built
  - ./configure –enable-cli –enable-ftd ….

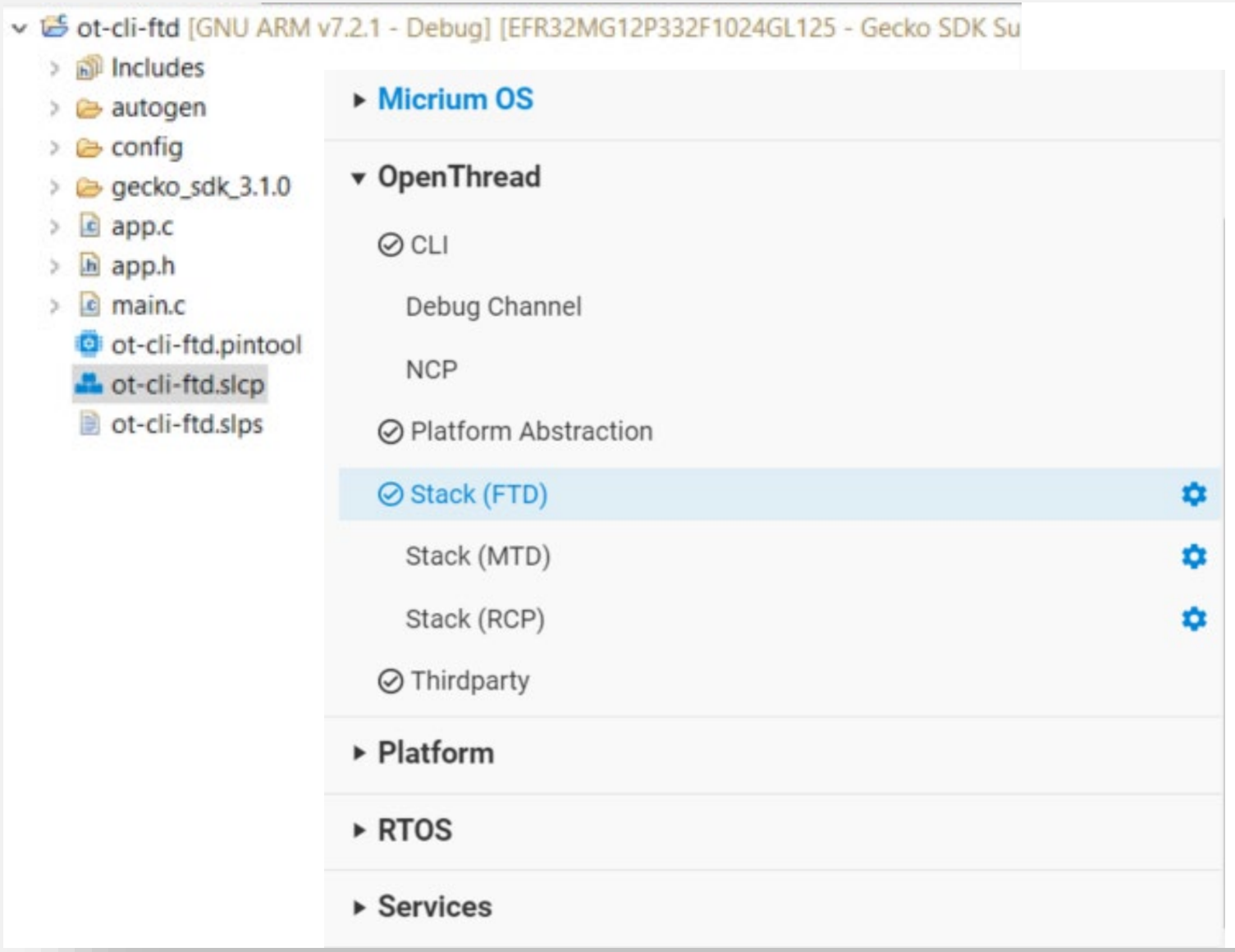
OpenThread within GSDK

Motivations

- Maintainability - SDK versions
- Extendibility - Hardcoded board support, hardcoded part support
- Redundancy - Repeated code across the Abstraction Layer
- Extensibility - Support Silabs features (Power manager, NVM3)

Solution: Integrating OpenThread with our GSDK!

OpenThread and UC



- Integration with SL tech.
  - NVM3
  - DMP
- Metadata from UC
  - For board specific configuration
- Components
  - Used to be plugins.
  - Can have requirements
  - Generation is automatic / makes sure requirements are satisfied
- Third party dependencies
  - mbedtls

OpenThread GSDK Integration (overview)

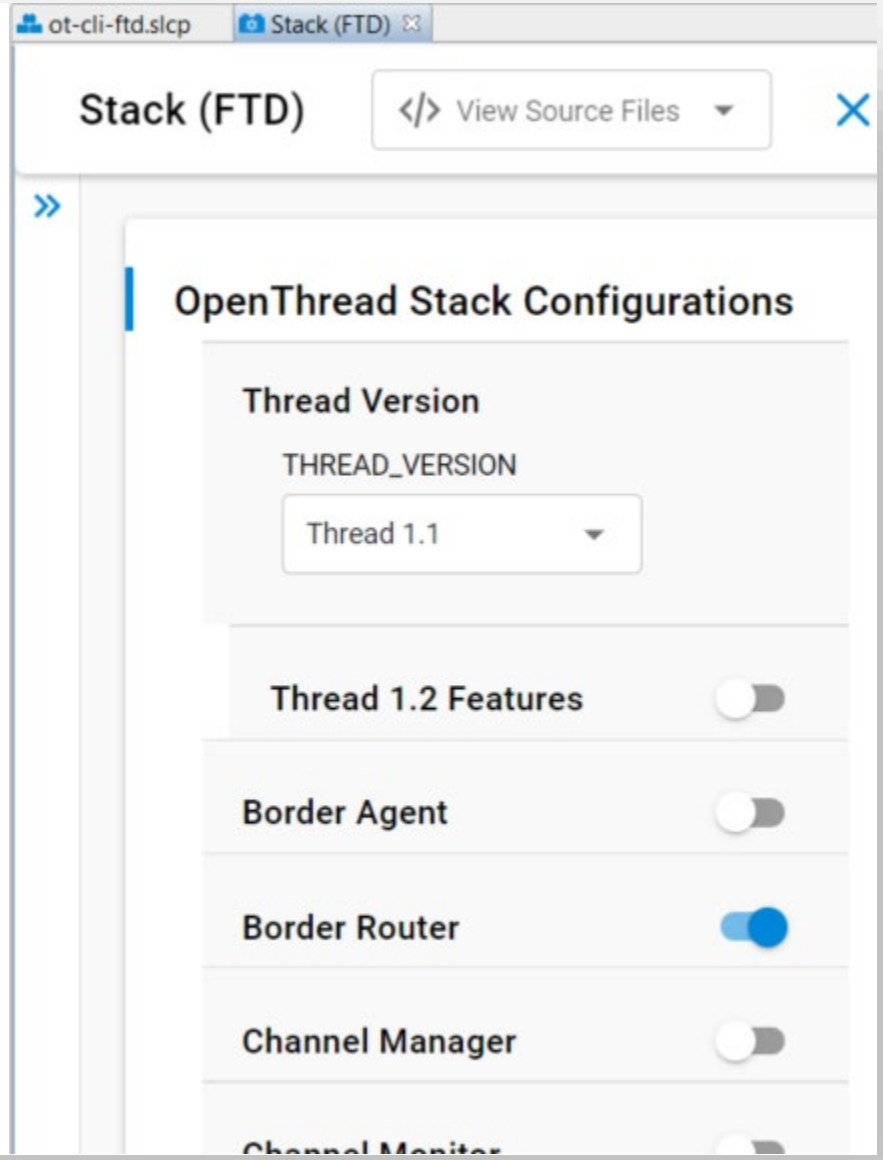
- OpenThread GSDK ⇔ (exact snapshot) OpenThread GitHub
- What's exactly same? Core stack implementation

`root_path: util/third_party/openthread/src/core`

- What's different? Platform Abstraction Layer (PAL)

`protocol/openthread/platform-abstraction/efr32/`

- Why is it different?
  - Needed a PAL that would fit in the GSDK 3.0 system
  - Avoid part specific code, clean legacy stuff



OpenThread GSDK version management

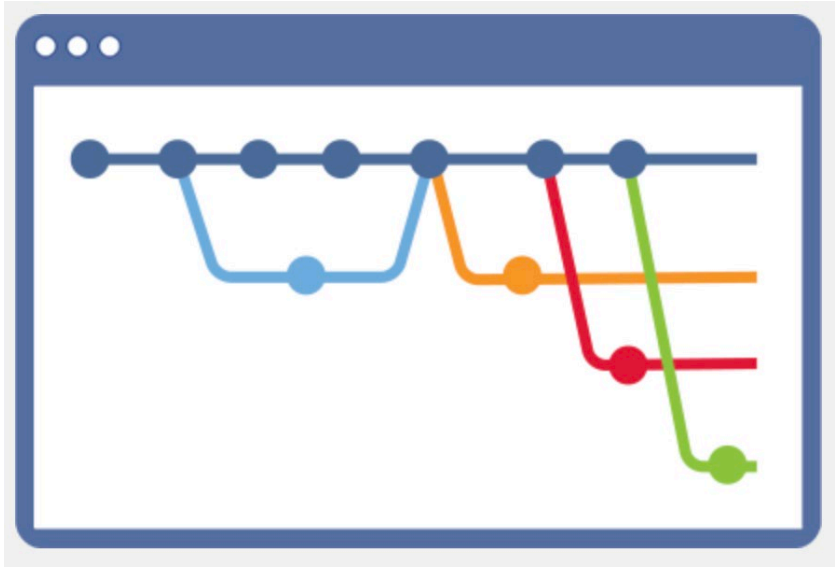
OpenThread Version Updates in GSDK:

- Updated bi-weekly
- Locked in May for the 20Q2
- Every Release has a commit hash to identify the snapshot

Gecko SDK Suite: Micrium OS Kernel, OpenThread 1.1.0.0 (GitHub-5c2ad91cf), Platform 3.0.0.0

Testing

- SQA: CI jobs for testing Supported parts
- PA: Functional tests. Sample apps.
- GRL test harness: For Thread 1.1 certification



Going forward?

- Will be updated regularly
- However, will need to be synced up with the version of ot-br-posix!

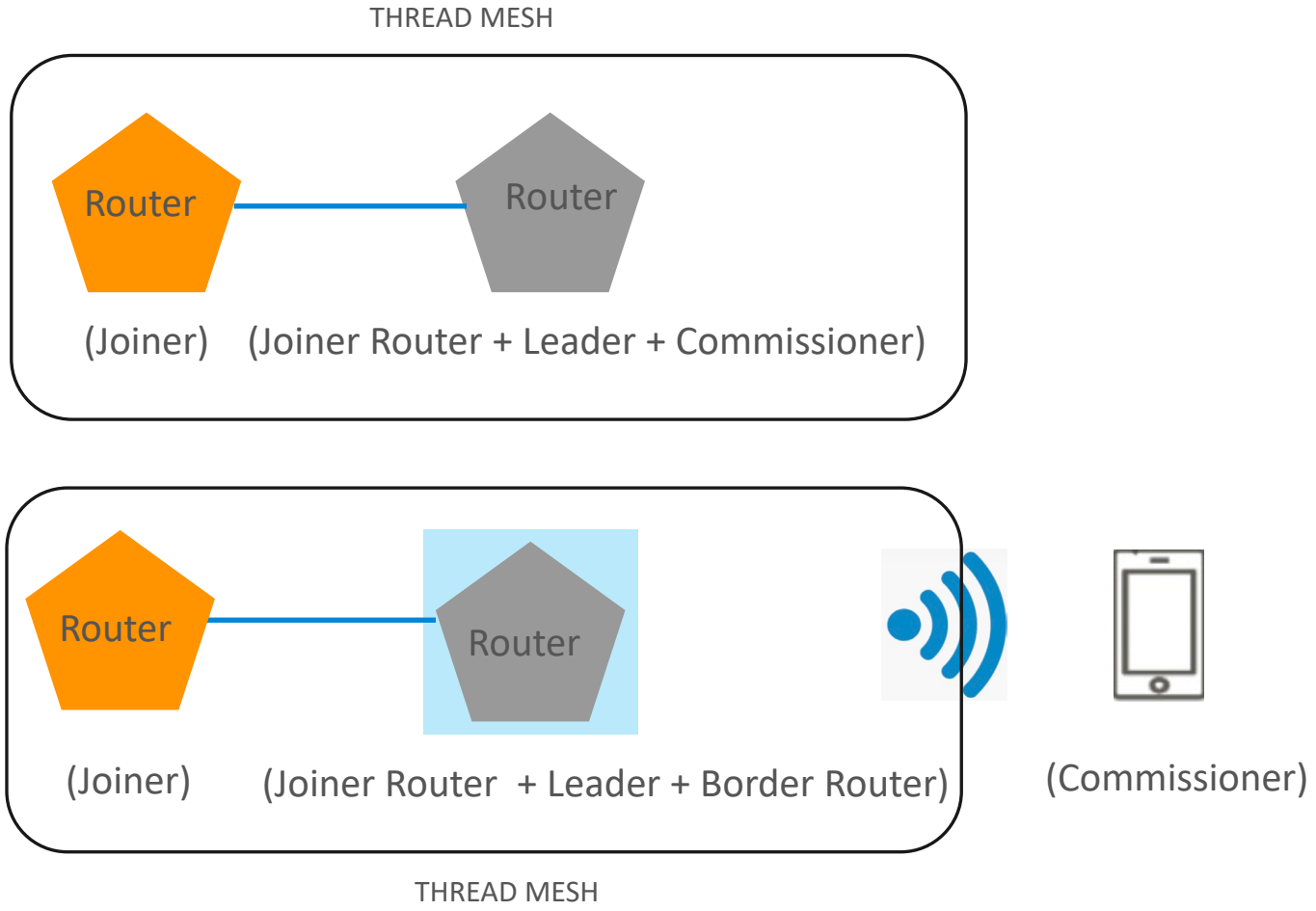
Comissioning

Commissioning: On-mesh vs Off-mesh

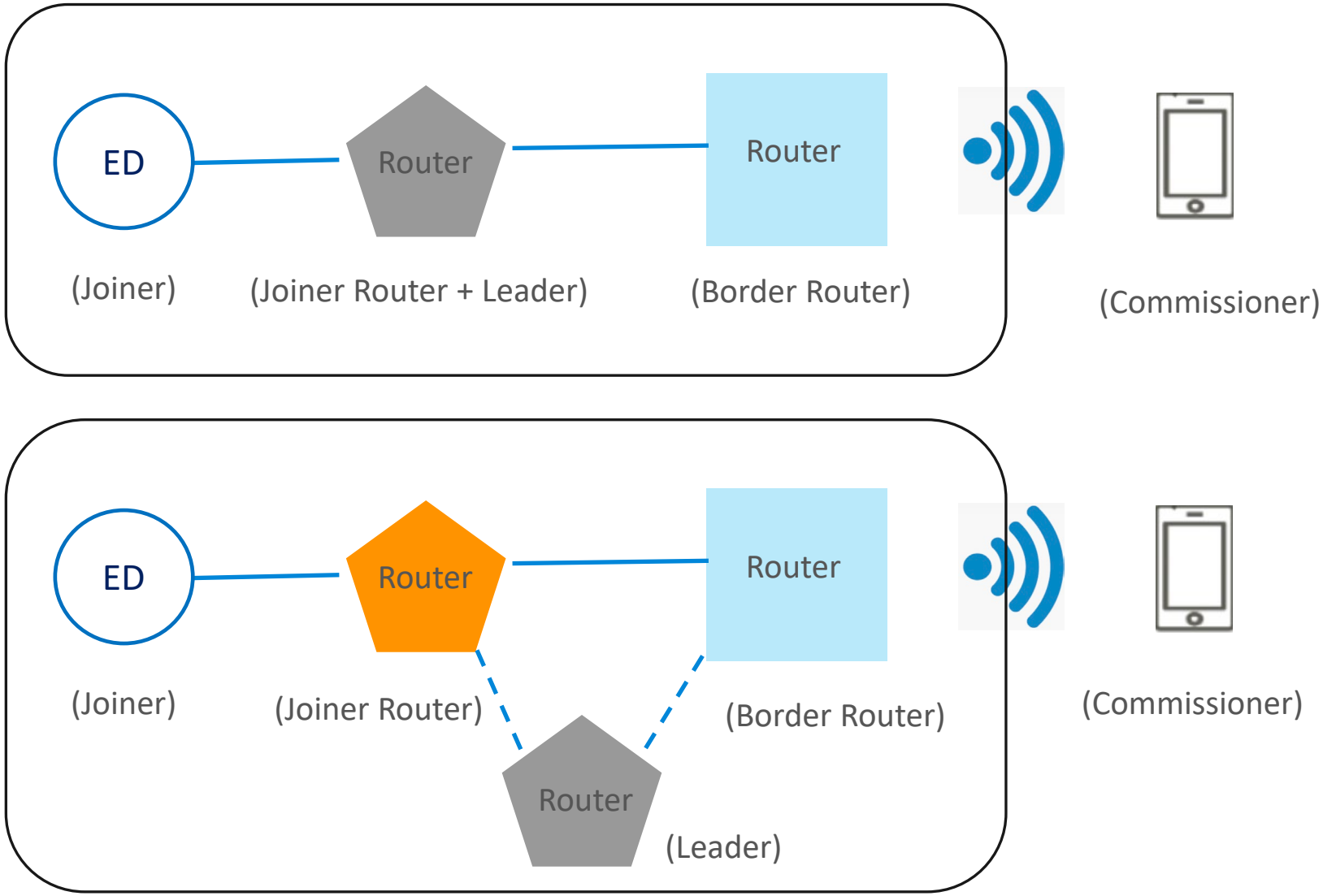
Typical flow:

1. Form a network (will need Border Router capability for Off-mesh case)
2. Configure a commissioner
3. Configure commissioner with allowed EUIs, Joiner Phrase (+ PSKc external)
4. Start join process

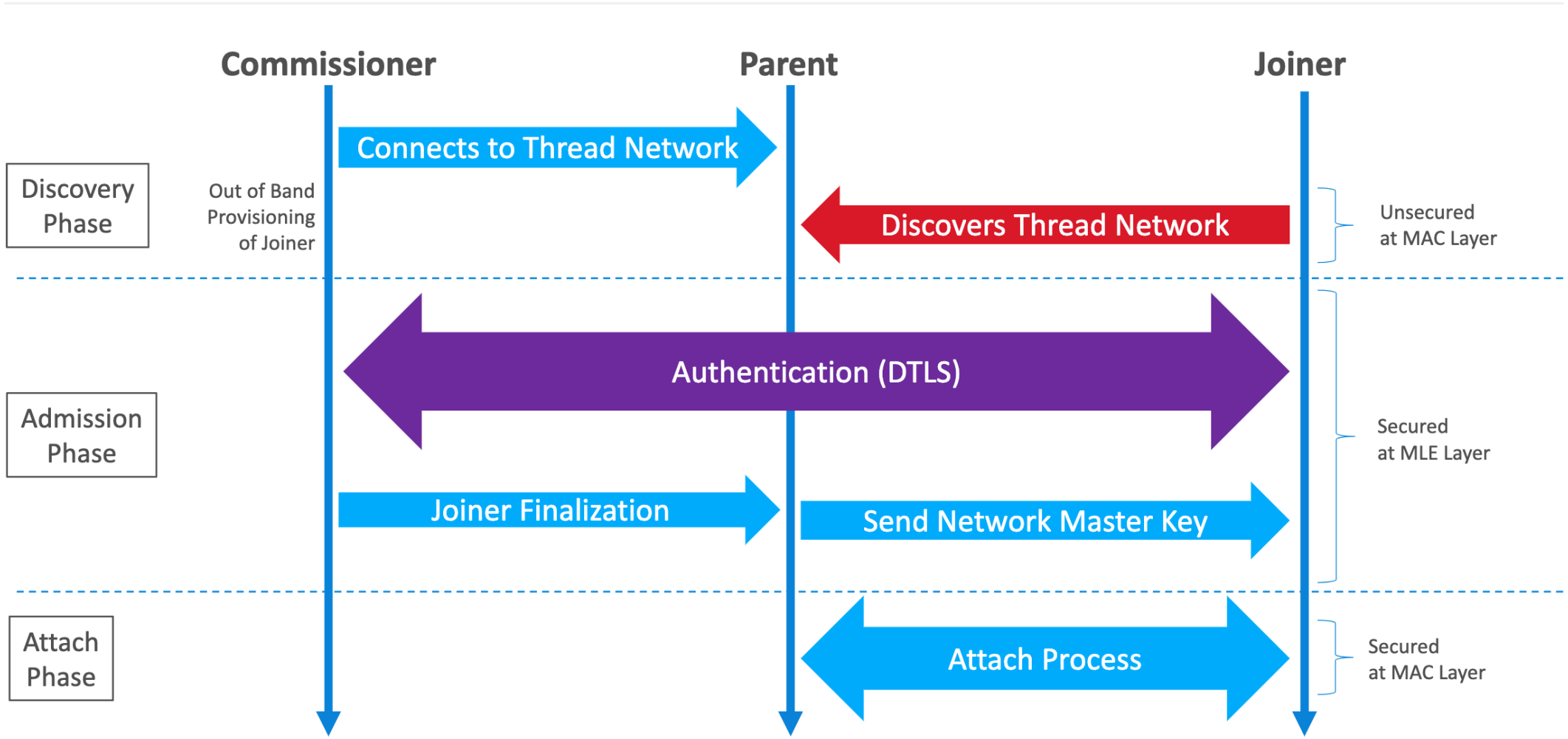
- How does the network learn about a BR / Commissioner? *Network Data TLV!*
- Who manages it ? *The Leader!*



Off mesh commissioner + Border Router



In-Band Commissioning – Message Exchange



In-Band Commissioning: Network Analyzer Trace

Transactions total:15 shown:15

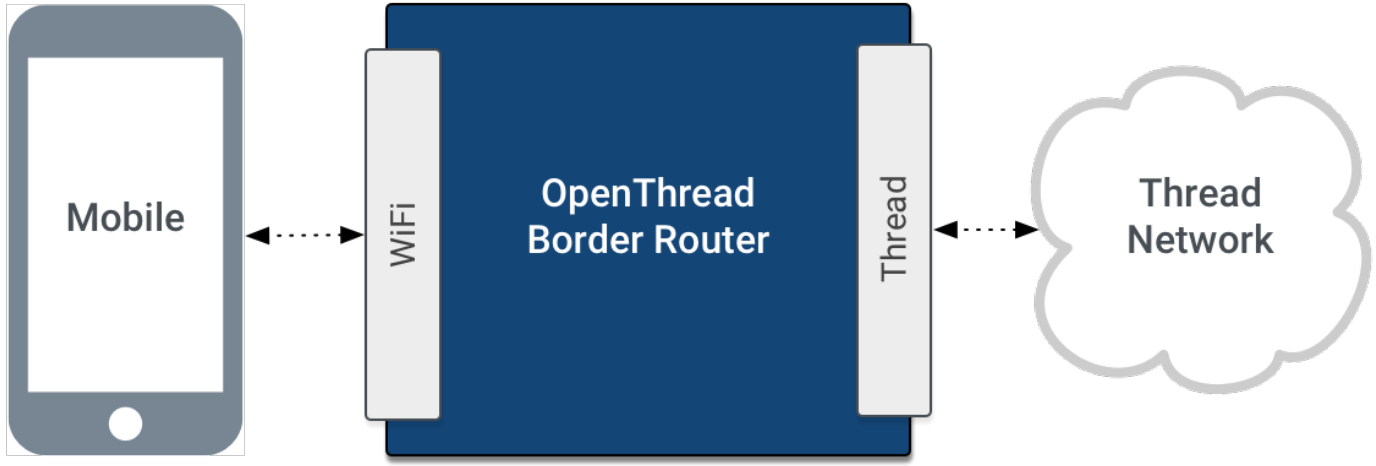
Time	Duration	Summary	IPv6 Src	IPv6 Dest	P#
3,950920	0,003	MLE Discovery Response	fe80::dcab:d8f2:eb96:b5...	fe80::407:8dc0:8638:cefb	2
5,058700	0,033	DTLS:hs[client_hello]	fe80::f108:9b30:a458:53c2	fe80::dcab:d8f2:eb96:b5...	10
5,316463	0,004	DTLS:hs[client_hello_verify]	fe80::dcab:d8f2:eb96:b5...	fe80::f108:9b30:a458:53c2	2
5,322196	0,032	DTLS:hs[client_hello]	fe80::f108:9b30:a458:53c2	fe80::dcab:d8f2:eb96:b5...	10
5,907104	0,044	DTLS:hs[server_hello],hs[server_key_exchang...	fe80::dcab:d8f2:eb96:b5...	fe80::f108:9b30:a458:53c2	13
6,515037	0,017	DTLS:hs[client_key_exchange],change_ciphe...	fe80::f108:9b30:a458:53c2	fe80::dcab:d8f2:eb96:b5...	6
6,766863	0,004	DTLS:change_cipher_spec,hs[hello_request]	fe80::dcab:d8f2:eb96:b5...	fe80::f108:9b30:a458:53c2	2
6,780253	0,005	DTLS:application_data	fe80::f108:9b30:a458:53c2	fe80::dcab:d8f2:eb96:b5...	2
6,797488	0,003	DTLS:application_data	fe80::dcab:d8f2:eb96:b5...	fe80::f108:9b30:a458:53c2	2
6,805424	0,003	DTLS:content_alert	fe80::f108:9b30:a458:53c2	fe80::dcab:d8f2:eb96:b5...	2
6,814321	0,003	DTLS:content_alert	fe80::dcab:d8f2:eb96:b5...	fe80::f108:9b30:a458:53c2	2
6,819736	0,003	DTLS:content_alert	fe80::dcab:d8f2:eb96:b5...	fe80::f108:9b30:a458:53c2	2
20,686869	0,004	MLE Parent Response	fe80::dcab:d8f2:eb96:b5...	fe80::10a7:e9d5:ca32:cf0	1
21,004754	0,004	MLE Child ID Request	fe80::10a7:e9d5:ca32:cf0	fe80::dcab:d8f2:eb96:b5...	2
21,013017	0,011	MLE Child ID Response	fe80::dcab:d8f2:eb96:b5...	fe80::10a7:e9d5:ca32:cf0	3

Border Router

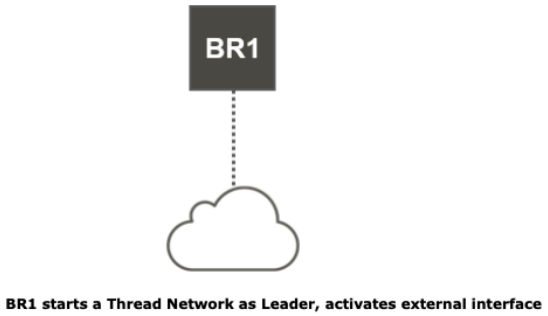
Characteristics

Border Router:

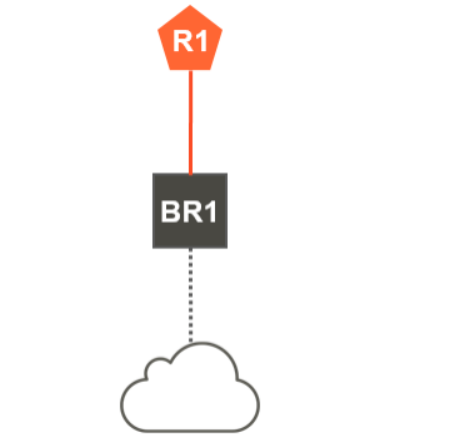
- Has **both 802.15.4 and IP link-layer interface** (WiFi or Ethernet)
- Performs IP routing
  - From Thread to Outside
  - From Outside to Thread
  - Can filter packets
  - Participate in external Routing.
- Transparent to end-to-end IP comms
- Should enable the Commissioner.
- May provide optional App Layer services.



On mesh (Thread network) role

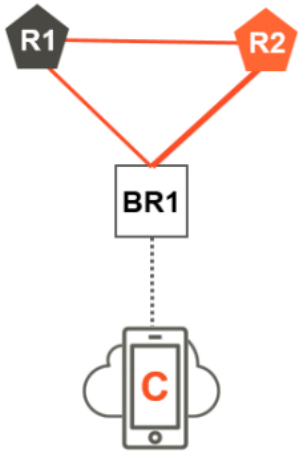


BR1 starts a Thread Network as Leader, activates external interface



BR1 acts as an internal Commissioner for joining R1

BR1 becomes unavailable. R1 takes over Leader role.



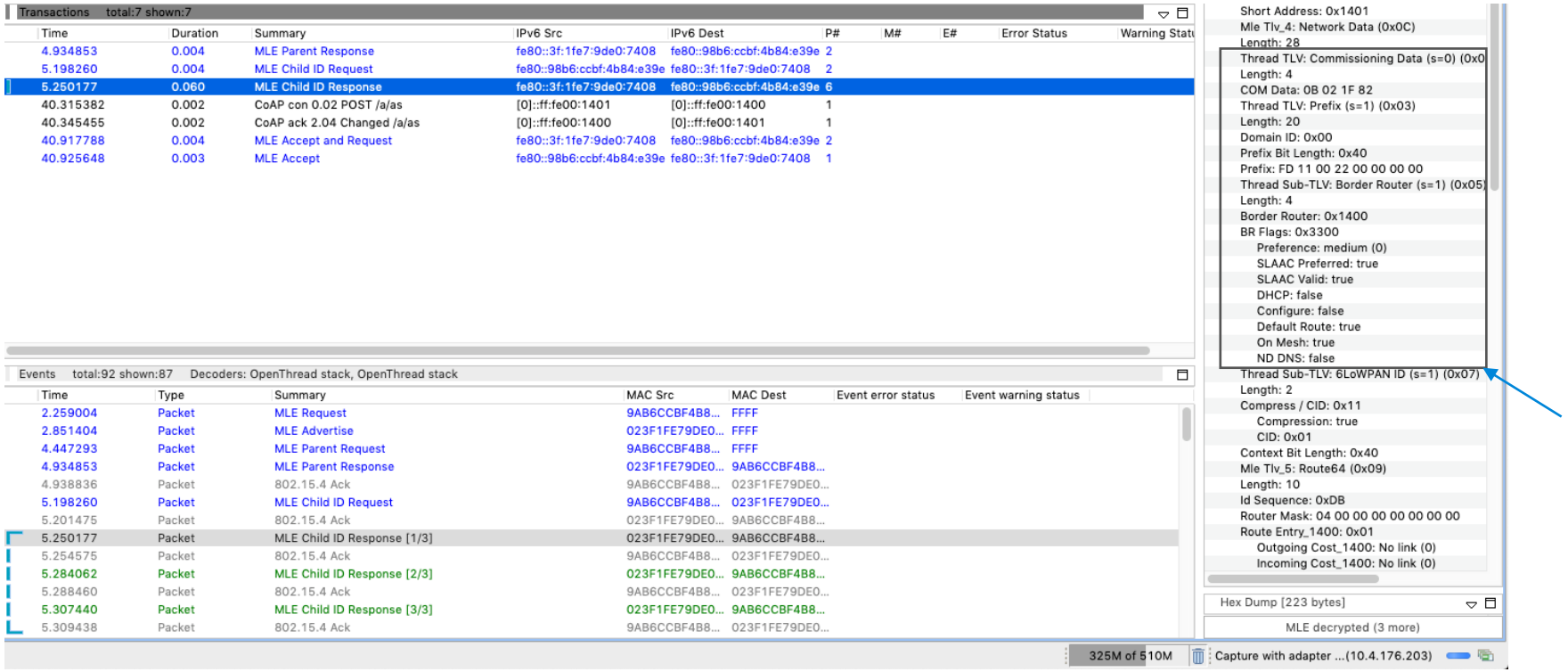
BR1 returns to the network. R1 remains Leader.

On a Thread Network, a Thread Border Router:

1. Can act as a Thread leader.
2. Can act as an on-mesh commissioner. (Discuss with Leader)
3. Must serve network data to the external network prefixes.
4. Should offer a prefix for global address configuration of Thread devices.
5. Should contact Leader about Thread NWK data changed.
6. Has to do Thread routing.

(+1). Backbone Router. (Thread 1.2 only)

On mesh (Thread network) role



What are the various IPv6 addresses of a Thread node?

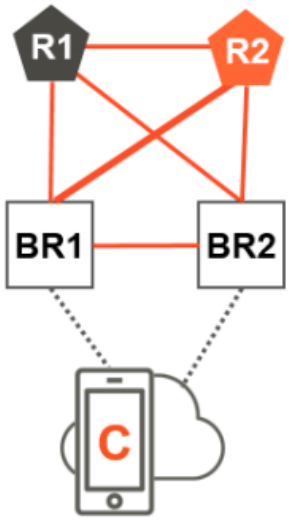
Example (using OT commands):

- Network is formed with a mesh-local prefix **`fdde:ad00:beef::/64`**
- Border Router offers a global routable prefix **`fd11:22::/64`** with the following properties: [SLAAC, On-mesh prefix, Default route, Stable, Preferred]

Consequently, every device on the network should have addresses that look as follows (example):

```
> ipaddr
fd11:22:0:0:6f6c:3683:774c:f281       global unique address
fdde:ad00:beef:0:0:ff:fe00:1401       rloc16
fdde:ad00:beef:0:d411:3e0e:7c31:83a   mesh local address
fe80:0:0:0:98b6:ccbf:4b84:e39e        link local address
Done
```

Off mesh role

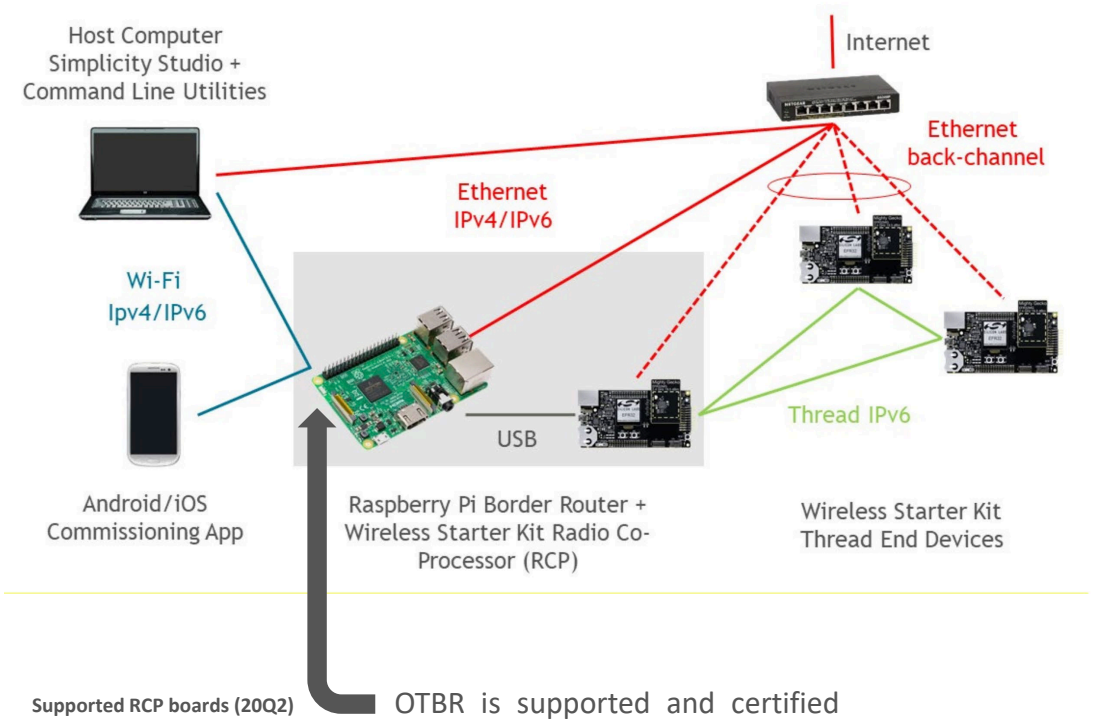


BR2 joins the network and takes over SLAAC role. BR1 and BR2 provide external routing.

Outside a Thread Network, a Thread Border Router:

1. Must implement IP layer packet forwarding between the Thread interface and exterior interface
2. Can perform exterior routing, neighbor discovery, address translation
3. May advertise global IPv6 prefixes, and act as a proxy for service discovery on behalf of the Thread network
4. Backbone Router – only in Thread 1.2

OpenThread Border Router Features



Supported RCP boards (20Q2)

Board Name
brd4161a
brd4166a
brd4168a
brd4180a
brd4304a

OTBR is supported and certified as a Thread component on a Raspberry Pi 3B host platform
(It can also run in a Docker container on a Raspberry Pi 3B or any Linux platform)

Github repo: https://github.com/openthread/ot-br-posix

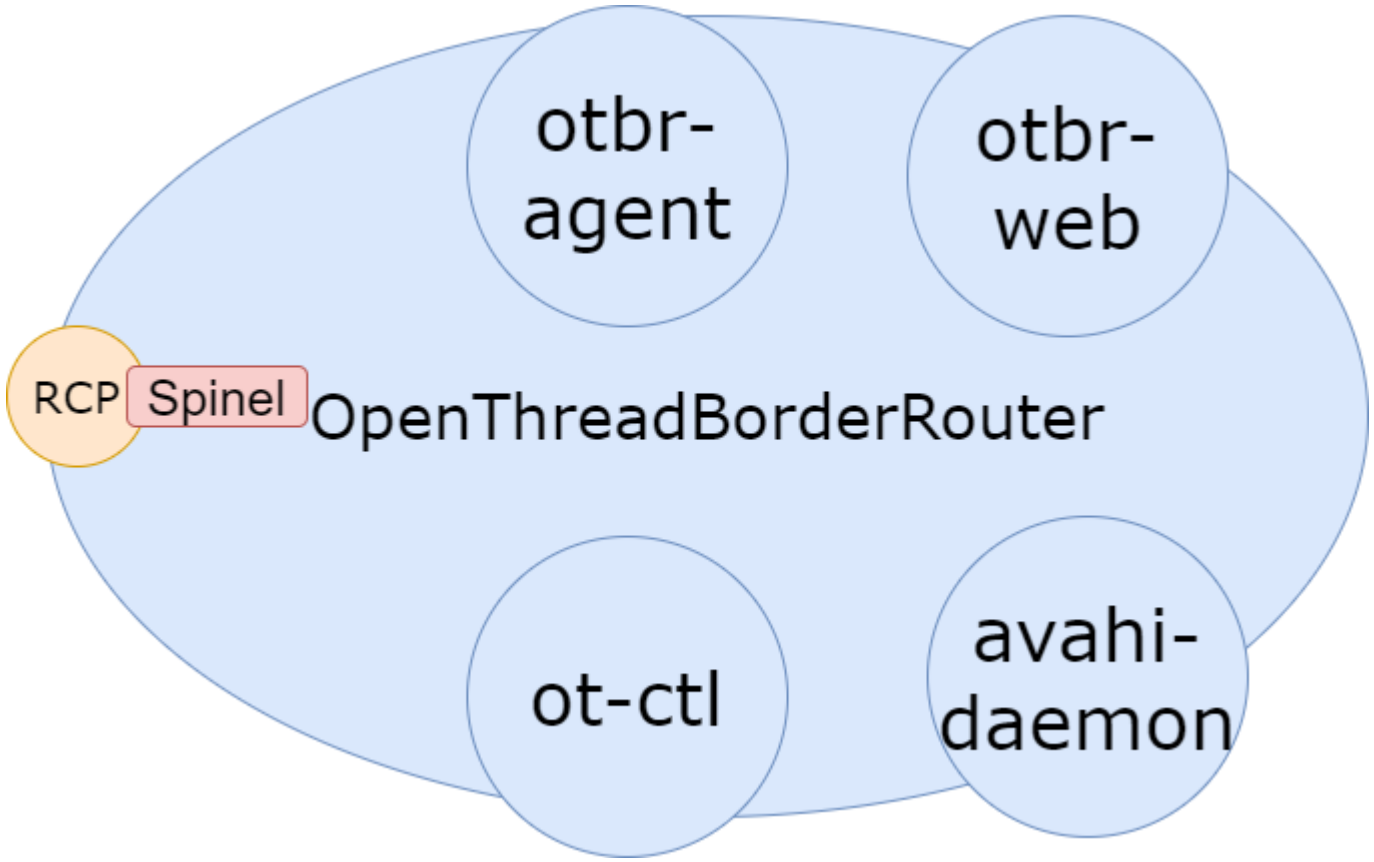
Primary source of documentation: https://openthread.io/guides/border-router

OTBR features

1. Thread Border Agent that can support external commissioning
2. DHCPv6 Prefix Delegation to obtain IPv6 prefixes for a Thread network
3. NAT64 for connecting to IPv4 networks
4. DNS64 to allow Thread devices to initiate communications by name to an IPv4-only server
5. Docker support
6. Command line tool to communicate with and manually configure an attached IEEE 802.15.4 radio co-processor (RCP)
7. Web UI for configuration and management
8. Additional tools: mDNS publisher, PSKc generator, etc.

OpenThread Border Router Components

- **Unix process check: “ps” – “systemctl”**



Docker / Manual Install

Docker Install

- Advantages of running OTBR in a Docker container:
  - Ease of deployment
  - Ease of configurability, migration, reproducibility
  - Easy recovery from failure
  - Easy updates (drop-in / roll-back new containers)
- Long steps in documentation or misconfiguration could negatively impact user experience.
- The deployed applications / containers can be easily run through test suites that verify that they operate correctly before promotion as production images (even if manual install is eventually desired)

Container based solution (Docker)

- Use a container image library like Docker Hub to publicly share deployable containers (***Hosting costs to be considered***)

AND

- Let customers build and deploy their own container using the Dockerfile provided in the repo (in GSDK)

AND

- Let them modify existing Dockerfile(s) or layer multiple Dockerfiles if necessary in a custom production environment

Manual Install

- Some customers might desire more granular control, moving to a customized process after prototyping with Docker. We would still need to support them.
- In some cases, containerized solutions are not supported by specialized network applications (for example, the Thread 1.1 commissioner and test harness do not support a Docker solution)
- Dev, QA and Support can still understand what is happening under the hood

Manual install

- Direct access to border router applications on platform/ Mostly same as 20Q2, keep up and document changes as when pulled from github

AND

- (Maybe) Provide a pre-configured ISO, or multiple ISOs for various versions

AND

- (Eventually) Support customers intending to run a border router on platforms other than a raspberry pi

Thank You for paying attention! Questions?