FTF 2016
TECHNOLOGY FORUM

HANDS-ON: THREAD STACK ADVANCED CLASS

FTF-HMB-N1961

CRISTIAN COTIGA, SOFTWARE PRODUCT MANAGER
ANTHONY HUERECA, SYSTEMS ENGINEER
FTF-HMB-N1961
MAY 18, 2016

AGENDA

- General Thread Introduction
- Thread Networking Architecture
- Thread Modules and their Roles
- Thread Security
- Thread Commissioning
- NXP Thread Offering
- NXP Thread Border Router Options
- NXP Thread Low Power End Devices
- Hands-On: Adding an I2C Sensor to a Kinetis Thread Router Eligible Device

GENERAL THREAD INTRODUCTION

Accelerating Technology Trends Drive Opportunities

Secure Connections for a **Smarter World**

Everything
Connected

1B+ additional consumers online, **30B+** connected devices

Connectivity

Everything
Smart

40B+ devices with intelligence shipped in **2020**

Processing

Everything
Secure

Potential savings to economy up to **half trillion dollars**

Security

Source: Euromonitor; Gartner; ARM Holdings; UBS; Center for Strategic and International Studies; McAfee, NXP analysis, International Telecommunications Union

Explosive Growth of Smart, Connected Solutions

SMART HOME

MCU
MPU
ANALOG
SENSORS

RF
NFC
STANDARD
PRODUCTS

SMART HEALTHCARE

MCU
MPU
ANALOG

SENSORS
NFC
STANDARD
PRODUCTS

SMART INDUSTRY

MCU
MPU
ANALOG
SENSORS

RF
NFC
STANDARD
PRODUCTS

WEARABLES

MCU
MPU
SENSORS

ANALOG
STANDARD
PRODUCTS

SMART INFRASTRUCTURE

MPU
Analog
RF

STANDARD
PRODUCTS

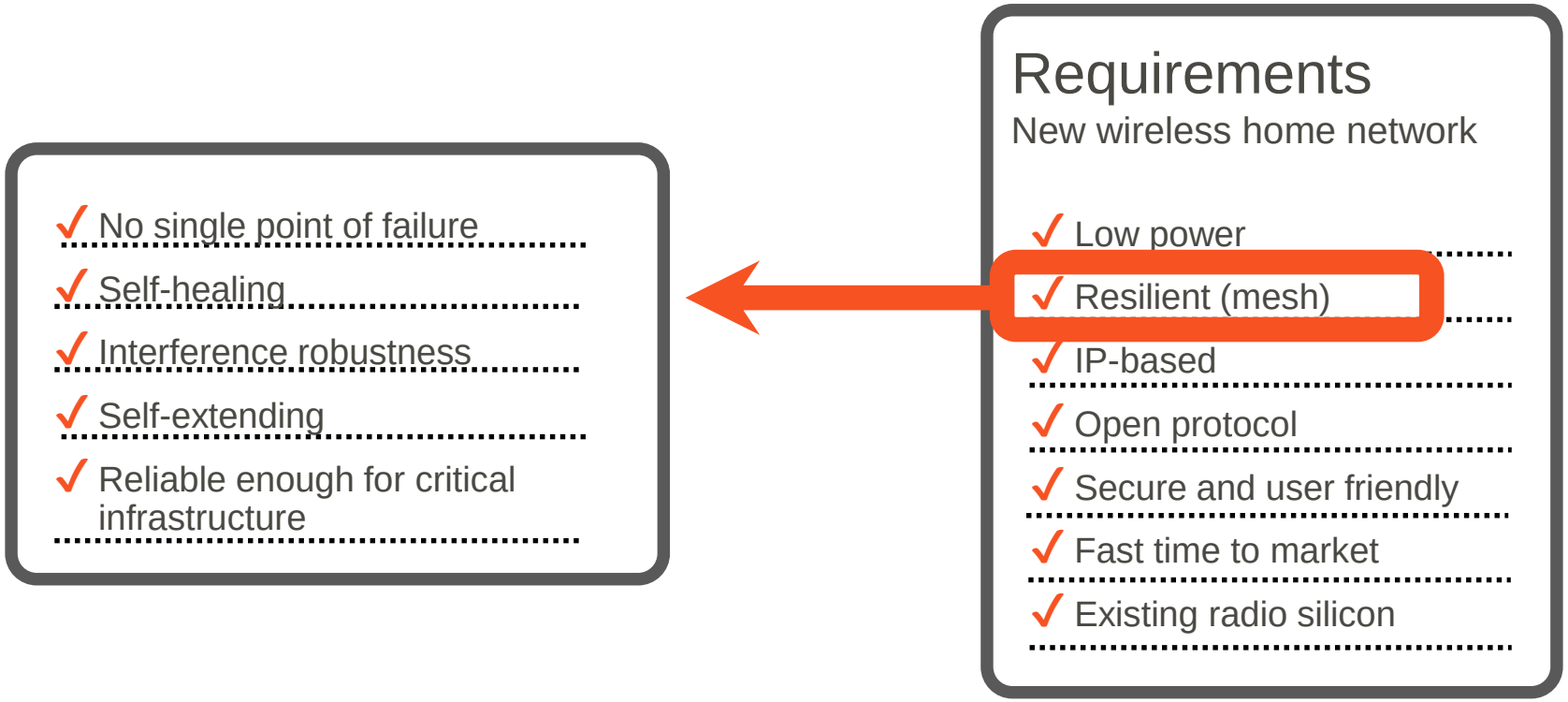
THREAD The need for a new wireless network

A new era of connected products

Existing wireless mesh protocol didn’t meet requirements

Other companies shared the same concerns

THREAD The need for a new wireless network



What is Thread?

A secure wireless mesh network for your home and its connected products

Built on well-proven, existing technologies

- Runs on existing 802.15.4 silicon
- Uses 6LoWPAN with IPv6 addressing
- UDP Transport

New mandatory security architecture

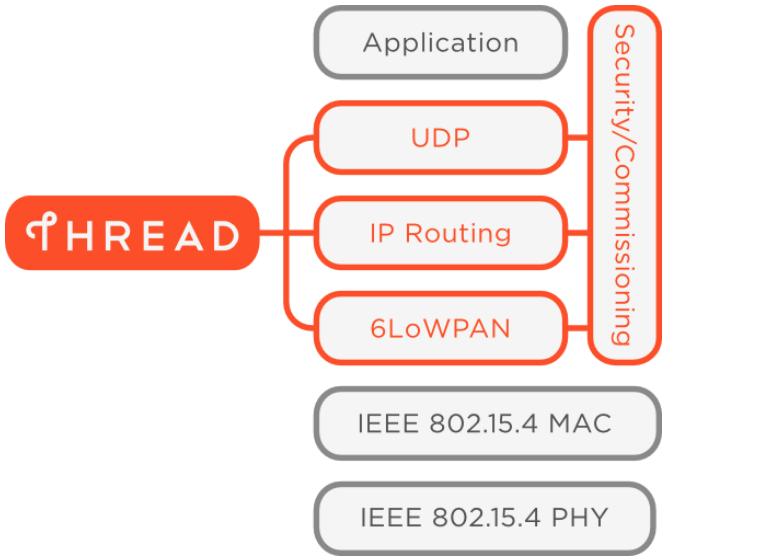
Simple and secure to add / remove products

Scalable to 250+ products per network

Designed for very low power operation

Reliable for critical infrastructure

Can support many popular
application layer protocols and platforms



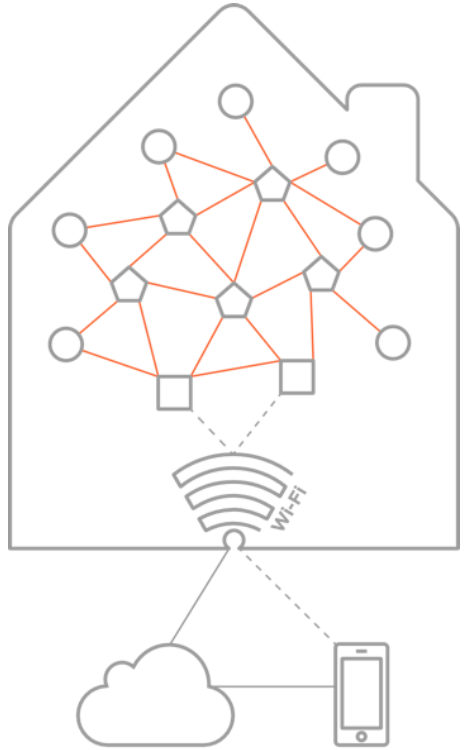
A software upgrade can add Thread to
currently shipping 802.15.4 products

Thread Specification is available to Thread Group members

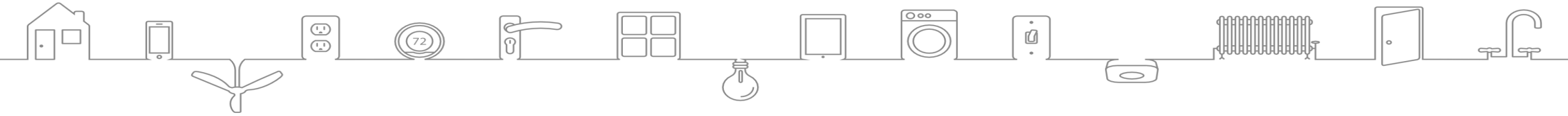
Target Applications

Thread is designed for all sorts of products in the home

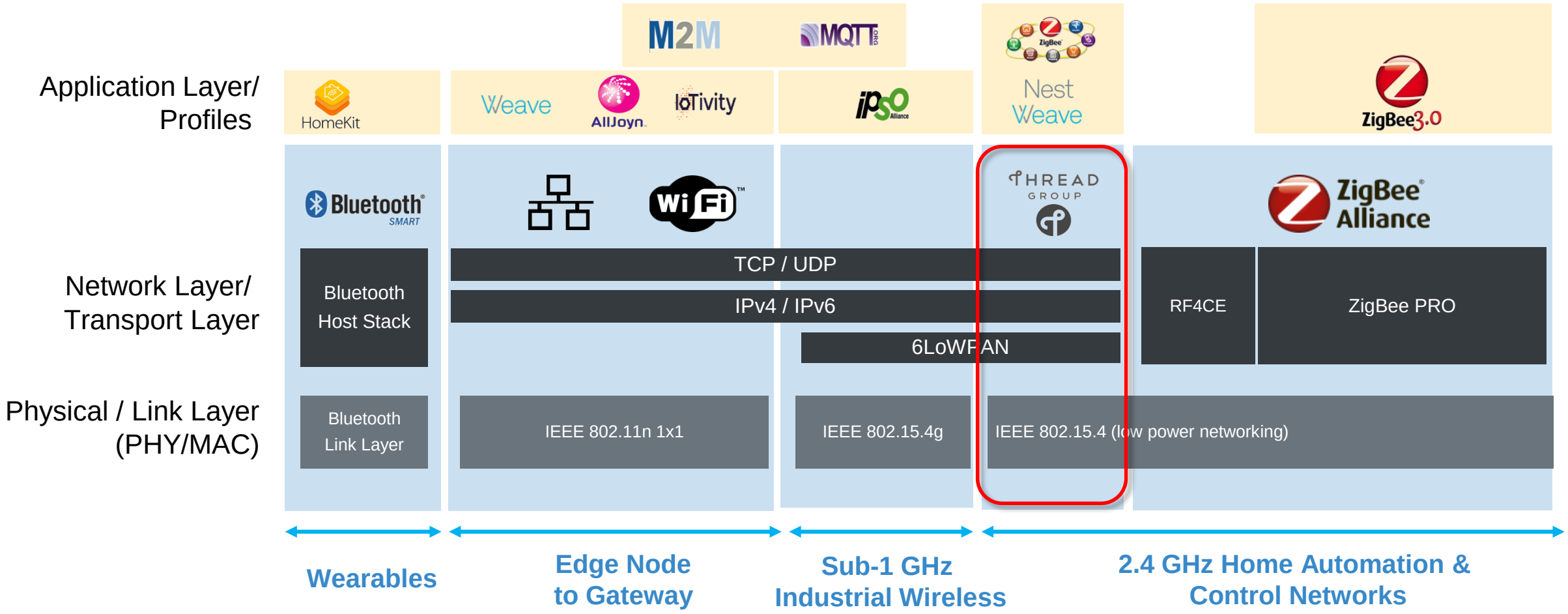
- Appliances
- Access control
- Climate control
- Energy management
- Lighting
- Safety
- Security



Devices working together to form a cohesive mesh network



IoT Connectivity Landscape – Where does Thread play?



Networking Technologies

	Wi-Fi 802.11 b/g/n	802.11ah (HaLow)	BT-LE 4.0	BT-LE SmartMesh v1	ZigBee Pro	Z-Wave Plus	Thread
Availability	LAN	✓	PAN	✓	WAN	✓	✓
Range / Topology	Long / Star	Long / Star	Short / Star	Short / Mesh	Short / Mesh	Short / Mesh	Short / Mesh
Topology	Star	✓	Star	Mesh	Mesh	Mesh	Mesh
No single point of failure	✗	✗		TBD	✗	✓	✓
Support for IPv6	✓	✓	Only 4.1+	✗	✗	✗	✓
Open Standards	✓	✓	✓	✓	✓	✗	✓
Multiple silicon vendors	✓	✓	✓	✓	✓	✗	✓
Application Layer	Multiple 3rd party options	Multiple 3rd party options	Multiple 3rd party options	Multiple 3rd party options	Native – ZCAL	Native	Multiple 3rd party options, Devices with different applicationss can still use each other for mesh communication
Use Cases / Benefits	Ubiquitous high-bandwidth wireless	Low power, long range, sub-gig	For devices tethered to your phone	Flood mesh, no support for IPv6, 10-byte payload	Purpose built end-to-end connectivity solution, Mission critical devices on own network		IP – based • Device-to-Device & Device to Cloud • Large base of IP-Developers Mission critical devices on own network, stable & secure for years

For comparison

About Thread Group

7 Founding Companies, grown to 11 Sponsor Companies

NXP founding company

A market education group offering product certification

Promoting Thread's use in connected products for the home

Thread will offer rigorous product certification to ensure security and interoperability

Board of Directors

President: Grant Erickson - Nest Labs

VP of Marketing: Sujata Neidig - NXP

VP of Technology: Skip Ashton - Silicon Labs

Secretary: Bill Curtis - ARM

Treasurer: Kevin Kraus - Yale Security

Director: Landon Borders – Haiku Home

Director: Christian Federspiel – OSRAM

Director: Rolf De Vegt - Qualcomm

Director: Mark Trayer - Samsung Electronics

Director: Cam Williams – Schneider Electric

Director: Jean-Michel Orsat - Somfy

Director: Greg Blackett – Tyco

Thread Membership as of Apr 2016

12

Sponsor companies

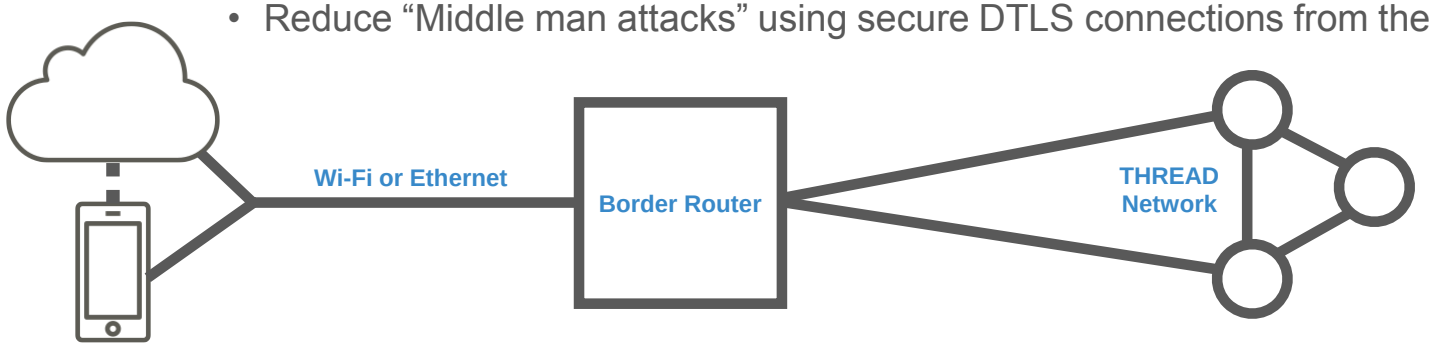
230+

Member companies

THREAD NETWORKING ARCHITECTURE

Promise of IoT requires IP All the Way to the End Node

- Cloud Services can address devices from the Internet
- Home Network can directly address devices through Border Routers
- Devices can address local devices on HAN or off network devices using normal IP addressing
- Reduce “Middle man attacks” using secure DTLS connections from the end node to the cloud



Cloud Connectivity

For control when not at home

When within the home, phone or tablet must go direct to gateway to eliminate latency of going to the cloud

Has to be seamless to consumer

Border Router

Bridge from the Thread Network to Wi-Fi/Ethernet

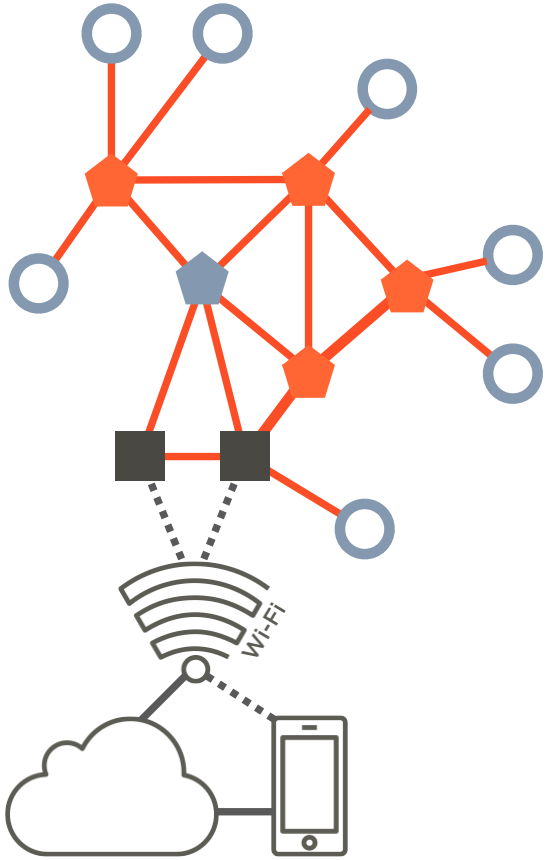
Forwards data to cloud

Provides Wi-Fi connectivity to phone, tablet or other devices in the home network.

Device Communication

Device to device communication within the Thread network for operations in the home

Network Architecture



End Device or Router Eligible Device

Active Router Eligible

Leader

Border Router

Thread Network

Network Topology Roles

Border Router

Border Router forwards data to and from the cloud

Also can provide Wi-Fi connectivity in the home

Thread Leader

Master of network parameters

Coordinates commissioners

Routes traffic among devices

Thread Active Router

Routes traffic among devices

Thread Routers form backbone of the Thread mesh

Leader-eligible

End Device

Designed for low power

May be powered or sleepy

May be router-eligible if powered

Many + One + Up to 31 + Up to 512 per Active Router

=

Thousands of Devices per Network (16k)

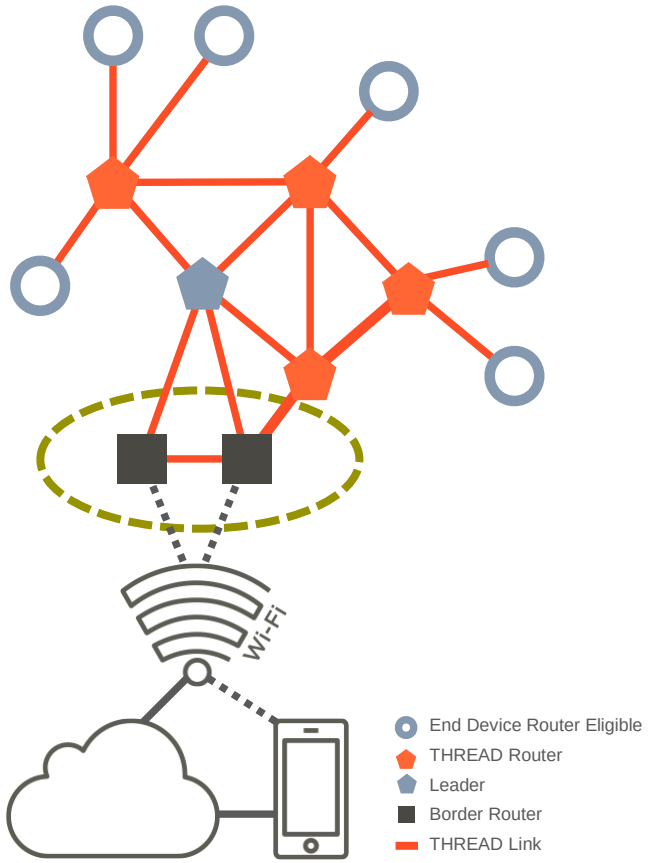
Kinetis Thread Stack – Number of Nodes per network

- **Nodes per network** = Active Routers + End Devices (Router Eligible End Devices, Powered End Devices, Sleepy End Devices)

- **Practical Limits:**
  - Practical limit of Children depends on Parent memory capacity
  - **Recommended number: 250 devices per subnet**

- **Kinetis Thread Stack:**
  - default settings for KW24/KW22:
    - **640 devices per subnet**
      - ▶32 Active routers
      - ▶20 Children per Parent
        - ▶could be increased with memory tradeoffs up to 50 children per active router

Thread Border Router



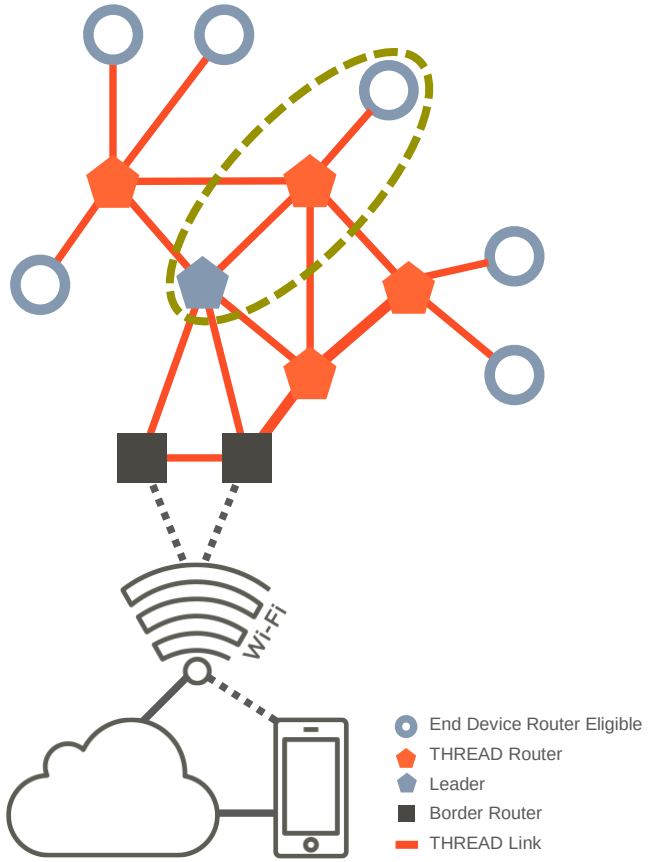
The **Border Router**

- Is usually a subset of Router Eligible Device
- Has at least one additional interface than IEEE 802.15.4 (e.g.: Wi-Fi, Ethernet, USB)
- Facilitates IP packet forwarding to and from the Thread network to home LAN or upstream IP infrastructure
- Can be multiple Border Routers in a Thread Network

The **Border Router**

- Can be a specialized networking device
  - Wireless Access Point (WAP)
  - Home Gateway
- Or can be embedded in a consumer product
  - Thermostat
  - Appliance

Thread Router Eligible Device



A **Router Eligible Device** can play multiple roles at runtime

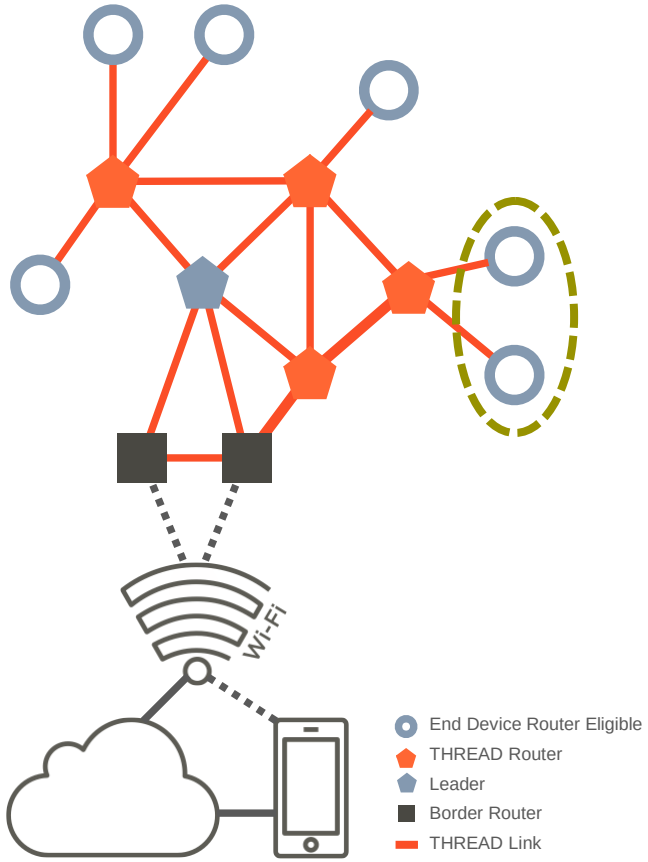
Leader
If it is the initial device in the network partition, or when the current leader is unavailable

Router Eligible End Device (REED)
Immediately after joining a network through an existing Active Routers or if the network has sufficient connectivity and does not need more routers

Active Router
A REED requests the Leader for it to become an Active Router when the network has relatively limited connectivity. e.g.: when total number of existing Active Routers is < 16

A **Router Eligible Device** is regularly a device meant to remain mains powered and **always on**

Thread End Devices



An **End Device**

- Does not have routing capabilities
- Communicates through a parent Active Router, but does not use data polling
- Cannot become a router (is not router eligible)

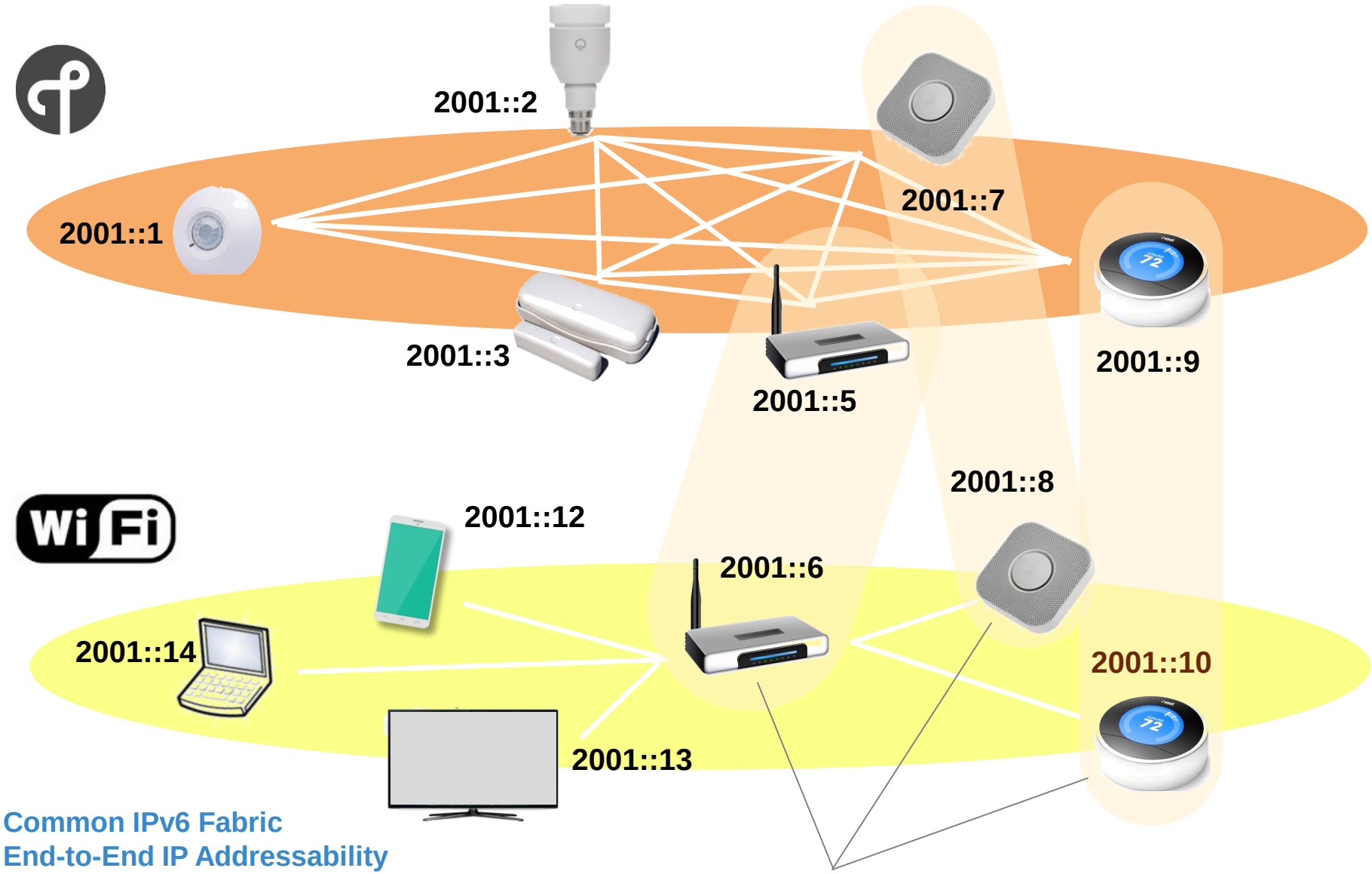
An **End Device** can be mains powered but **periodically turned off** or has a high capacity battery with recharge

A **Low Power End Device**

- Does not have routing capabilities
- Is a “Sleepy End Device” (SED), mostly having its radio transceiver turned off
- Communicates through a parent Active Router, and uses data polling to receive packets
- Cannot become a router (is not router eligible)

A **Low Power End Device** has a limited capacity battery, usually non rechargeable (e.g.: coin cell)

Thread and Wi-Fi: Common IP Infrastructure



So how many addresses does a Thread device get?

- Once joined to a network, a Thread device will get:

Thread internal only!!!

- **at least 3 Unicast IPv6 addresses to the Thread Interface:**

```
Link local address (LL64):                fe80::c180:1192:c3ea:ef55
Mesh local address (ML16, RLOC):          fda3:217b:338e::ff:fe00:400
```

OK to be used by application

```
Mesh local address (ML64, ML-EID):        fda3:217b:338e::213e:34b4:2659:10f8
```

- **two All Thread Nodes multicast addresses:**

```
Link local all Thread Nodes(Multicast):   ff32:40:fda3:217b:338e::1
Realm local all Thread Nodes(Multicast):  ff33:40:fda3:217b:338e::1
```

- **optionally will also get:**

```
Unique local address (ULA):               fd01::3ead:5d45:9bf0:94d5:4234
Global unique address (GUA):              2602:306:839b:bcca:8948:9563:2313:93cc
```

LL64 interface ID is based on IEEE 802.15.4 Random Extended Address

RLOC interface ID is based on IEEE 802.15.4 Short address (6-bits Router ID + Child ID)

Mesh Local Prefix is based on Extended PAN ID bytes 1-5

ML-EID interface ID is random

Mesh Local Prefix is based on Extended PAN ID bytes 1-5

Unique Local Prefix is defined in each Border Router

ULA interface ID is random

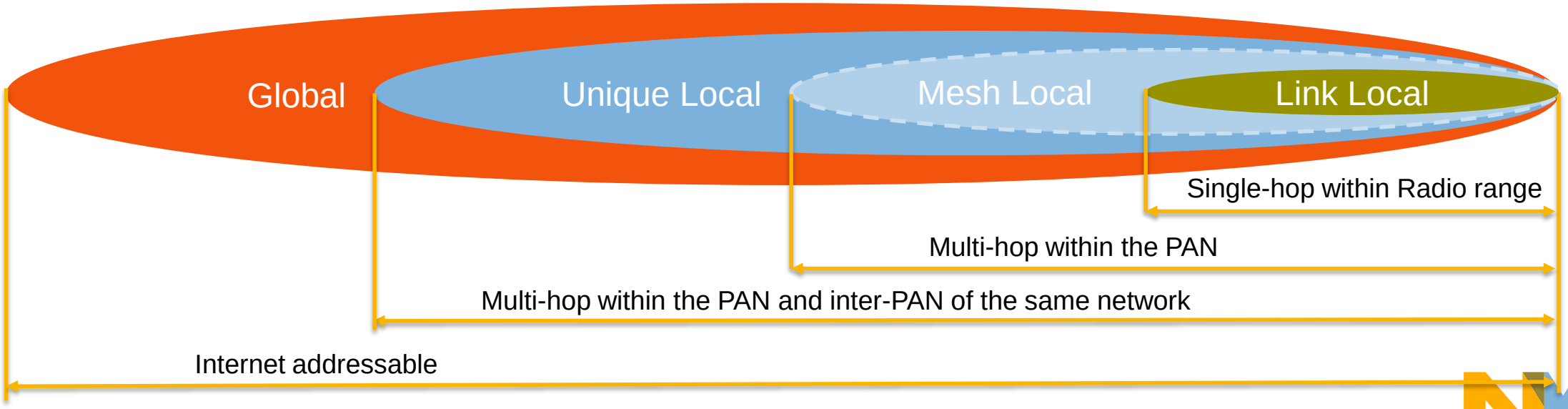
GUA interface ID assigned via DHCPv6 or random if assigned via SLAAC

Global Prefix retrieved from the ISP via prefix delegation

Use `ifconfig` command in Kinetis Thread Stack shell to obtain IP address configuration

IPv6 Unicast Address Scopes

- **Scopes** specify the boundaries of networks when using and forwarding packets for an address
- Defined by IANA: http://www.iana.org/assignments/ipv6-address-space/ipv6-address-space.xhtml
  - **Link Local Scope** – Addresses can be used only on the same network segment which shares link layer. Prefix: `FE80::/10`
  - **Unique Local Scope** – Addresses which are private to an administratively configured network and cannot be routed outside the network. Prefix: `FC00::/7` (`FC00::/8` and `FD00::/8`)
  - **Global Scope** – All other valid, non-reserved addresses. Prefix (currently): `2000::/3`



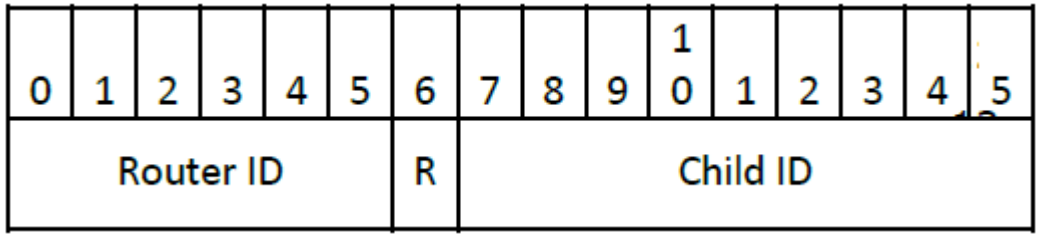
EIDs and RLOCs

- **EID: Endpoint Identifiers** – are Thread interface IPv6 addresses which:
  - Have a scope different than link-local (e.g.: Unique Local, or Global)
  - **Do not change** when the Thread mesh network topology changes, i.e. its assignment is tightly coupled to the device
  - The specific EID formed based of the 64-bit **Mesh Local Prefix** and a 64-bit random interface ID is the **ML-EID**, or **ML64** address and is automatically generated by the stack to be used by applications

- **RLOC: Routing Locator** – is a Thread interface IPv6 addresses which:
  - Is formed based on the **Mesh Local Prefix (ML16)** and the IEEE 802.15.4 short address of the device
  - **May change** when the Thread network topology changes because active routers short addresses get reassigned by the leader

IEEE 802.15.4 Short Addresses and RLOCs

- **IEEE 802.15.4 Short Address** format used to facilitate routing**:**



- 6 bits Router IDs: maximum 64 Router IDs can be assigned by the Leader Router to Active Routers
- 1 bit R is reserved and is set to 0
- 9 bits Child IDs: assigned by each Active Router (Child ID all 0s is reserved for the Active Router itself); Child IDs are used by REEDs, End Devices, Sleepy End Devices

- **RLOC Routing Locator IPv6 address (a.k.a. ML16 address):**

```
<Mesh Local Prefix>:0000:00FF:FE00:<IEEE 802.15.4 Short Address>
```

Thread IPv6 Multicast Scopes

- **Link Local scope** and **Realm-Local scope** multicast addresses are defined generically for IPv6 multicast for All Nodes `FF0x::1` and All Routers (`FF0x::2`)
- Assigned by IANA: http://www.iana.org/assignments/ipv6-multicast-addresses/ipv6-multicast-addresses.xhtml
- LL and RL scopes used in Thread networks for packet transmission based on the following rules:

`FF02::1`	All Neighbors but not forwarded to Low Power/Sleepy EDs
`FF02::2`	All Router Neighbors but not forwarded to Low Power/Sleepy EDs
`FF03::1`	All Nodes in Mesh but not forwarded to Low Power/Sleepy EDs
`FF03::2`	All Routers in Thread Mesh but not forwarded to Low Power/Sleepy EDs

- **All Thread Nodes multicast addresses** can be used for addressing all nodes, including Sleepy/Low Power End Devices

`FF32:40:ML::1`	All Neighbors single hop including Low Power/Sleepy EDs
`FF33:40:ML::1`	All Devices in the Thread network multi-hop including Low Power/Sleepy EDs

On-Mesh Prefixes

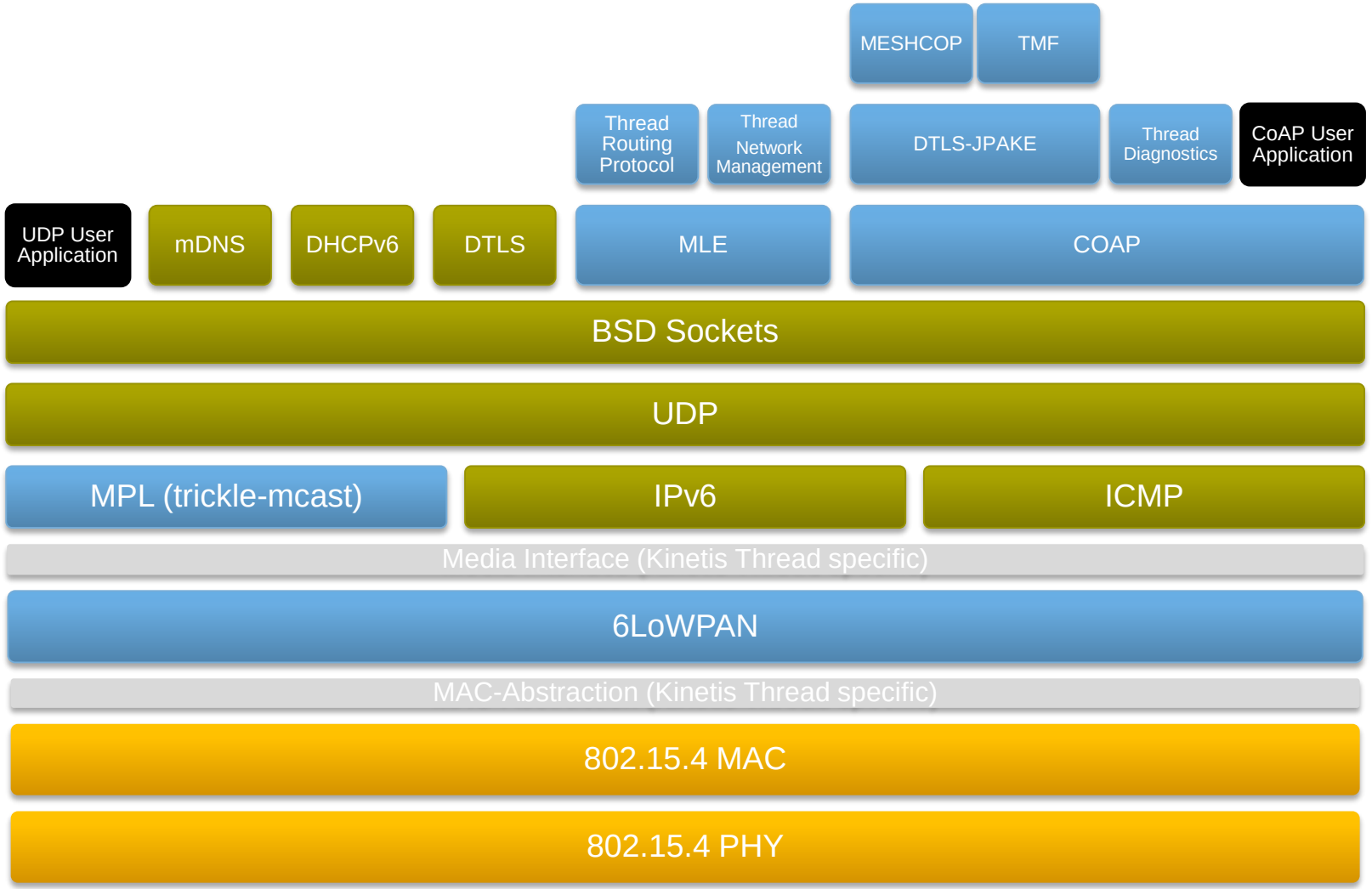
- **On-mesh prefixes are supplemental Unique Local or Global Scope prefixes advertised by servers on border routers via Network Data**

- Thread nodes can assign EID addresses using these prefixes using 2 methods:
  - **SLAAC – Stateless Address Auto-Configuration:** Interface ID is generated randomly and mapped to RLOCs through address query and address caching (similar to ML-EIDs).
  - **DHCPv6** – Require a DHCPv6 server running on a Thread node (usually on the border Router)

- Default route on-mesh prefix flag can be set administratively for directing outbound packets through certain Border Routers if there are multiple available

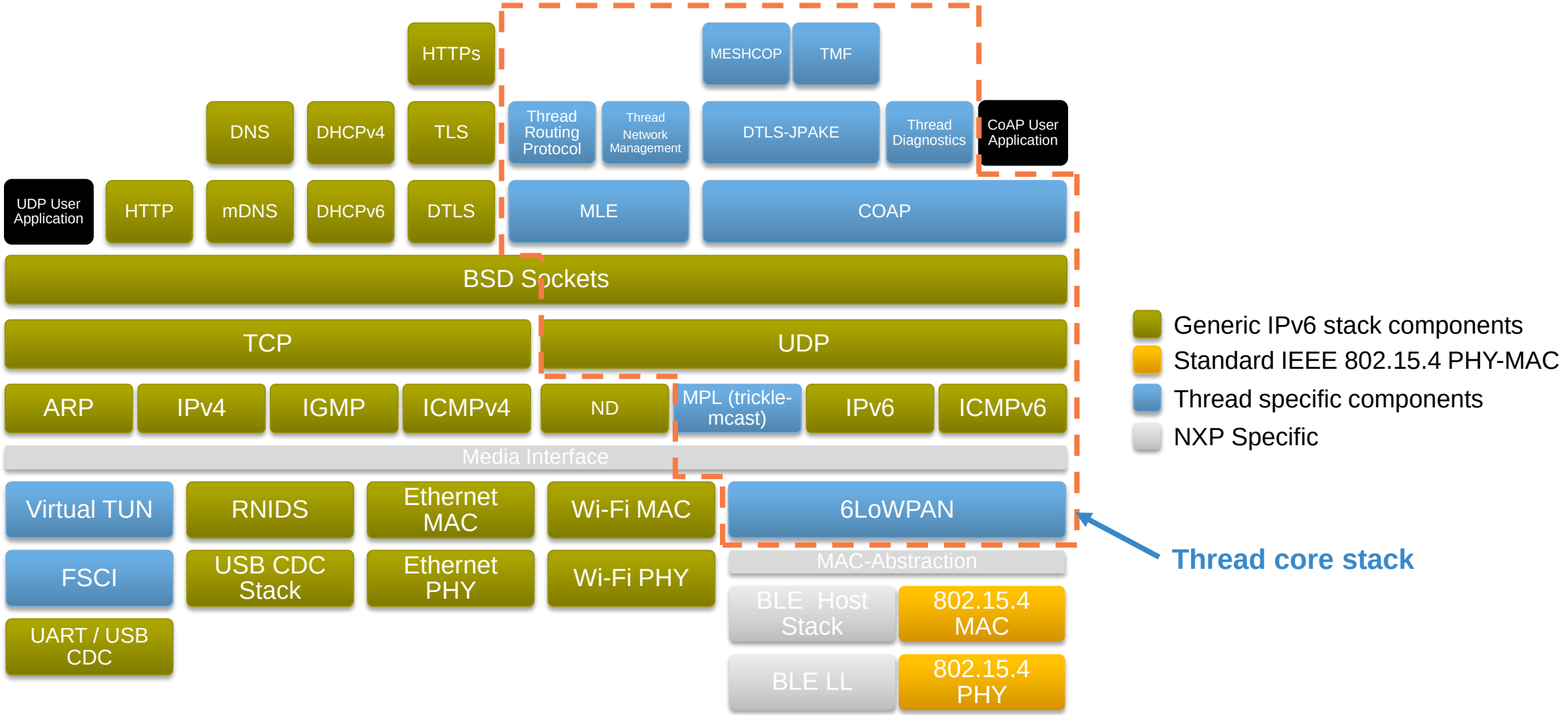
THREAD MODULES AND THEIR ROLES

Thread End Device - High Level Block Diagram



- Generic IPv6 stack components
- Standard IEEE 802.15.4 PHY-MAC
- Thread specific components
- NXP Specific

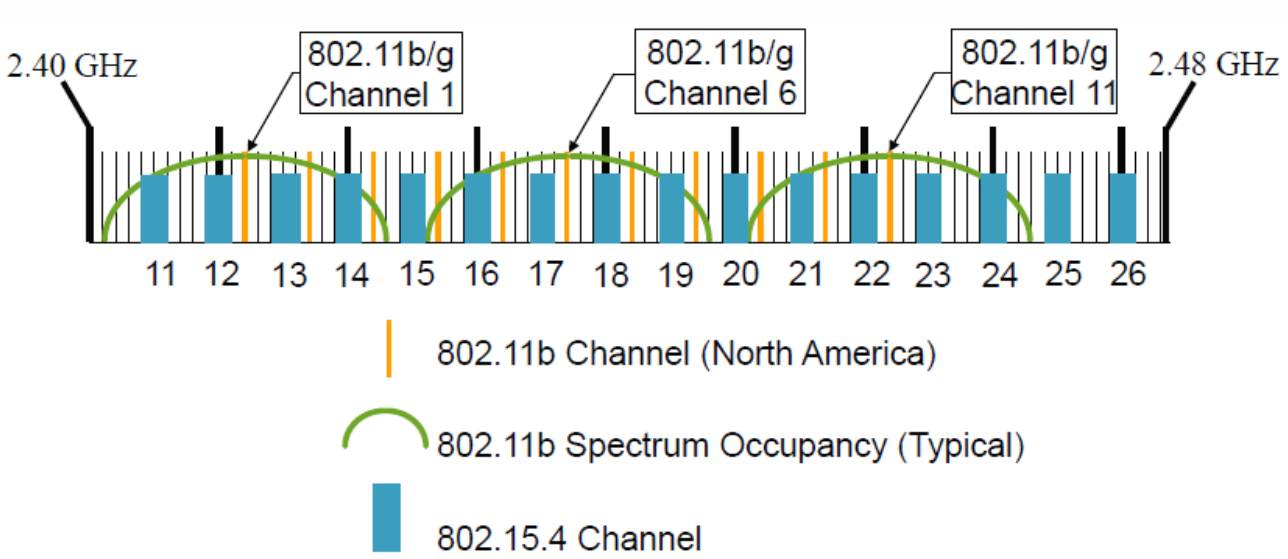
Thread Border Router – High Level Block Diagram

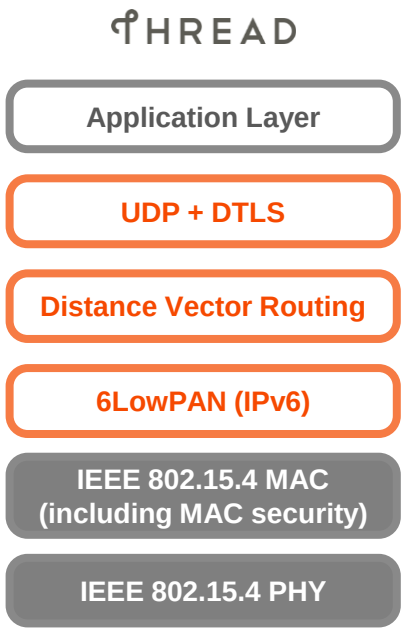


802.15.4 PHY-MAC

IEEE 802.15.4 - PHY

IEEE 802.15.4 channel occupancy on 2.4GHz





802.15.4 open channels when Wi-Fi fully utilized the band

- 15, 20, 25, 26.

IEEE® 802.15.4 MAC Functions

- Ensures reliable and secure data transfers
- Essential foundation for technologies like ZigBee® or Thread

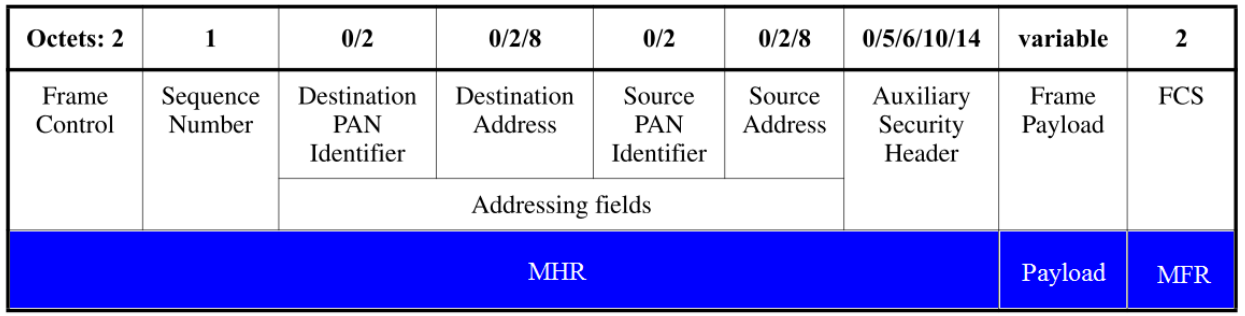
- Collision avoidance algorithm through clear channel assessment
- Acknowledgement-based transmissions and re-transmissions
- Integrity checks with CRC-16

- AES-128 data encryption and CCM* block ciphers authentication

- Allows star or peer-to-peer topologies
- IEEE® standard 64-bit or short, dynamic 16-bit addressing
- Dynamic device addressing allowing routed meshes in upper layers

- Optional slotted mode with superframe-based duty cycles
- Device segregation based on capabilities and roles in a network: coordinator and end device

IEEE® 802.15.4 MAC/PHY Frames

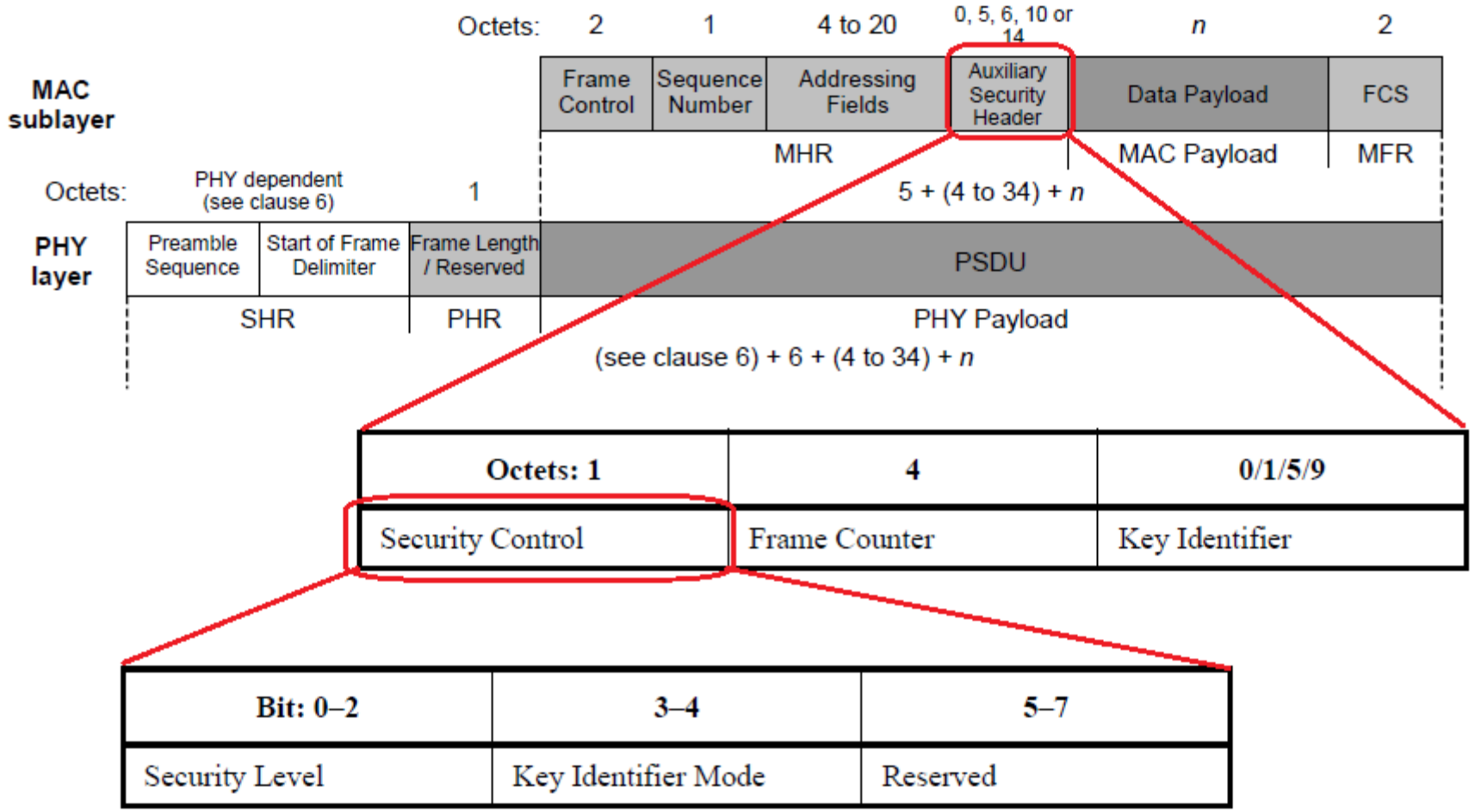


Octets: 2	1	0/2	0/2/8	0/2	0/2/8	0/5/6/10/14	variable	2
Frame Control	Sequence Number	Destination PAN Identifier	Destination Address	Source PAN Identifier	Source Address	Auxiliary Security Header	Frame Payload	FCS
		Addressing fields						
MHR							Payload	MFR

Four frame types:

- **Beacon**: Used for synchronization and broadcast of data
- **Data**: Used for data transmission
- **Acknowledgement**: Used to acknowledge data and command frames
- **MAC command**: Used to carry MAC management commands

IEEE® 802.15.4 MAC Secured Frames

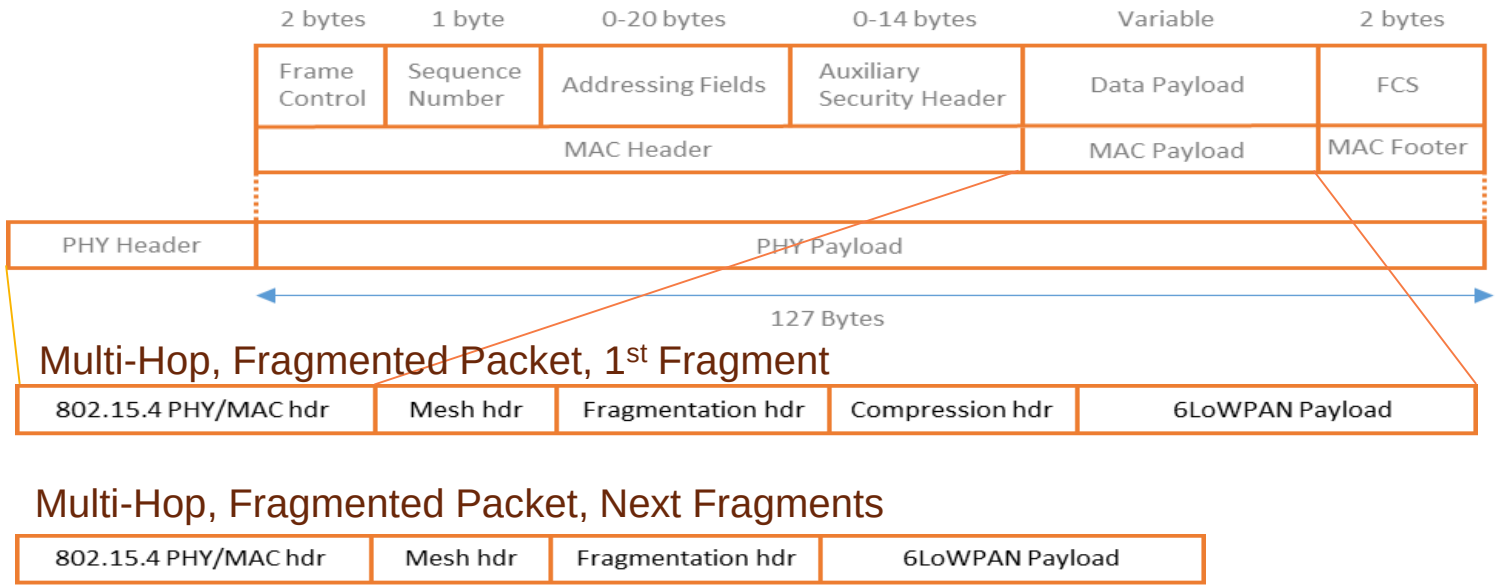


THREAD DATA PLANE

6LoWPAN

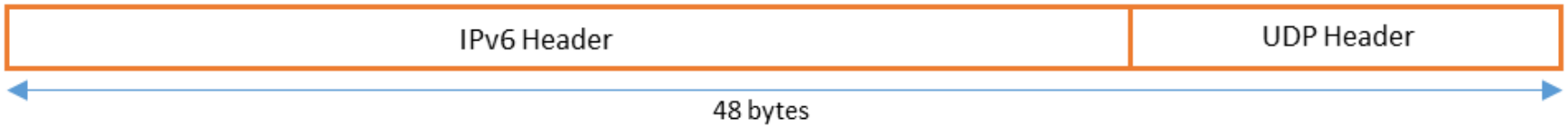
- **6LoWPAN** is an adaptation layer between the IEEE 802.15.4 MACPHY and IPv6 layer used as an IPv6 Media Interface within the constraints and requirements of both standards

- 6LoWPAN functionality in Thread is based on RFC 4944 and RFC 6282 and achieves the following:
  1. **IPv6 header compression** from 40+ bytes to <10 bytes
  2. **IPv6 packets fragmentation and reassembly** to / from smaller MAC-PHY payloads
  3. **IPv6 packets forwarding across multiple hops** using the mesh header

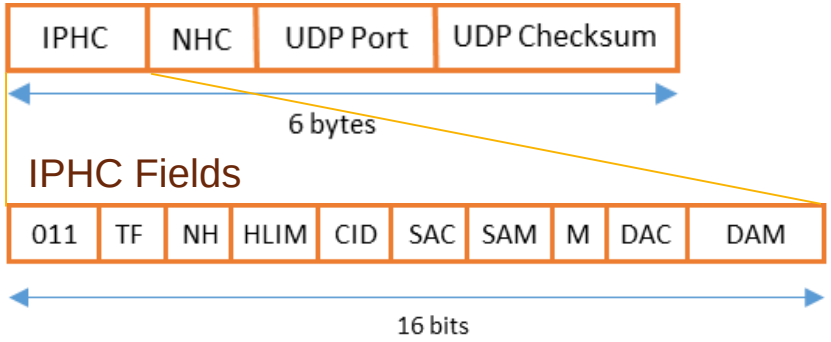


6LoWPAN Header Compression

Full IP Header



Compression Header (RFC 6282)

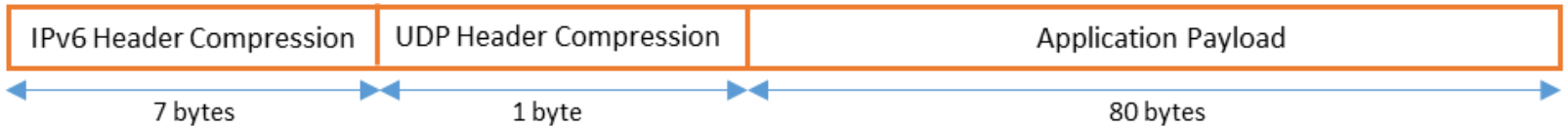


Header Compression reduces IPv6 and UDP headers from 48 to 6-8 bytes

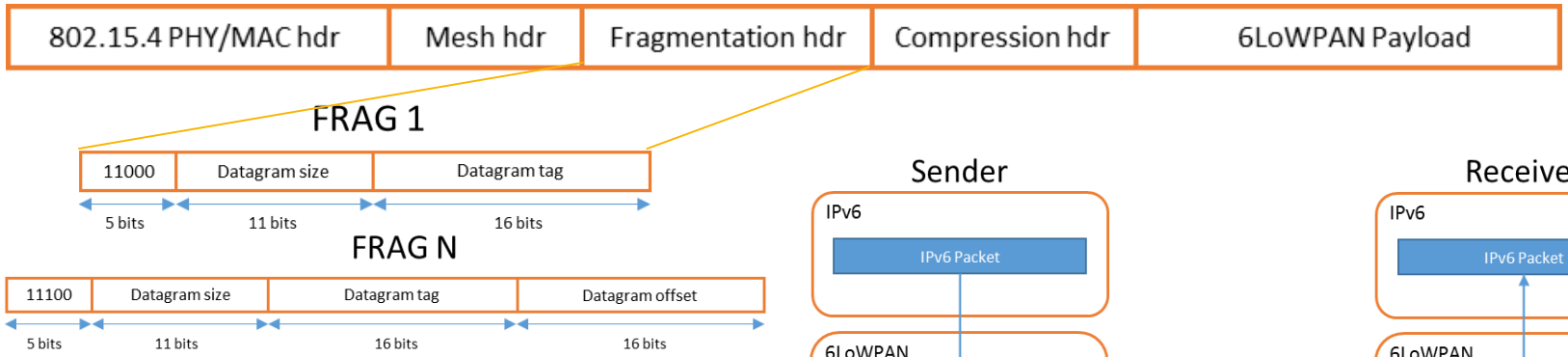
- Two headers get compressed: IPv6 header and UDP header
- IPv6 header compression
  - Stateless
    - Elides portions of source and destination IPv6 address based on the device IEEE 802.15.4 MAC addresses
  - Statefull (using contexts)
    - A 6LoWPAN Context represents an IPv6 prefix
    - For compression, the prefixes are elided and just the ContextID is sent instead
    - Prefixes are propagated through the network by the Thread Leader using MLE advertisements
- UDP header compression
  - Uses a 2 bits compression scheme for the most usual UDP ports
  - Elides the UDP checksum because IEEE 802.15.4 already has a checksum in the MAC frame

6LoWPAN Fragmentation

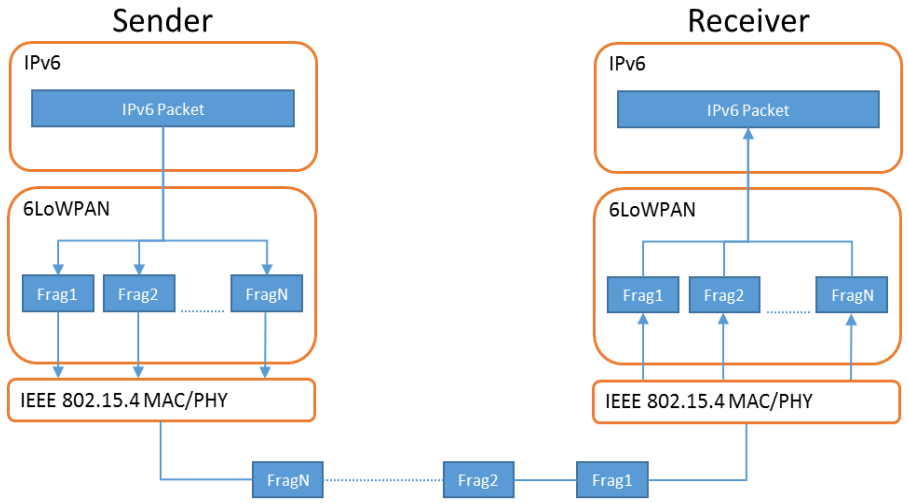
- Application payload with Header Compression is ~80 bytes per MAC frame; IP packet can be larger and is fragmented



- 6LoWPAN Fragmentation with Fragmentation Header

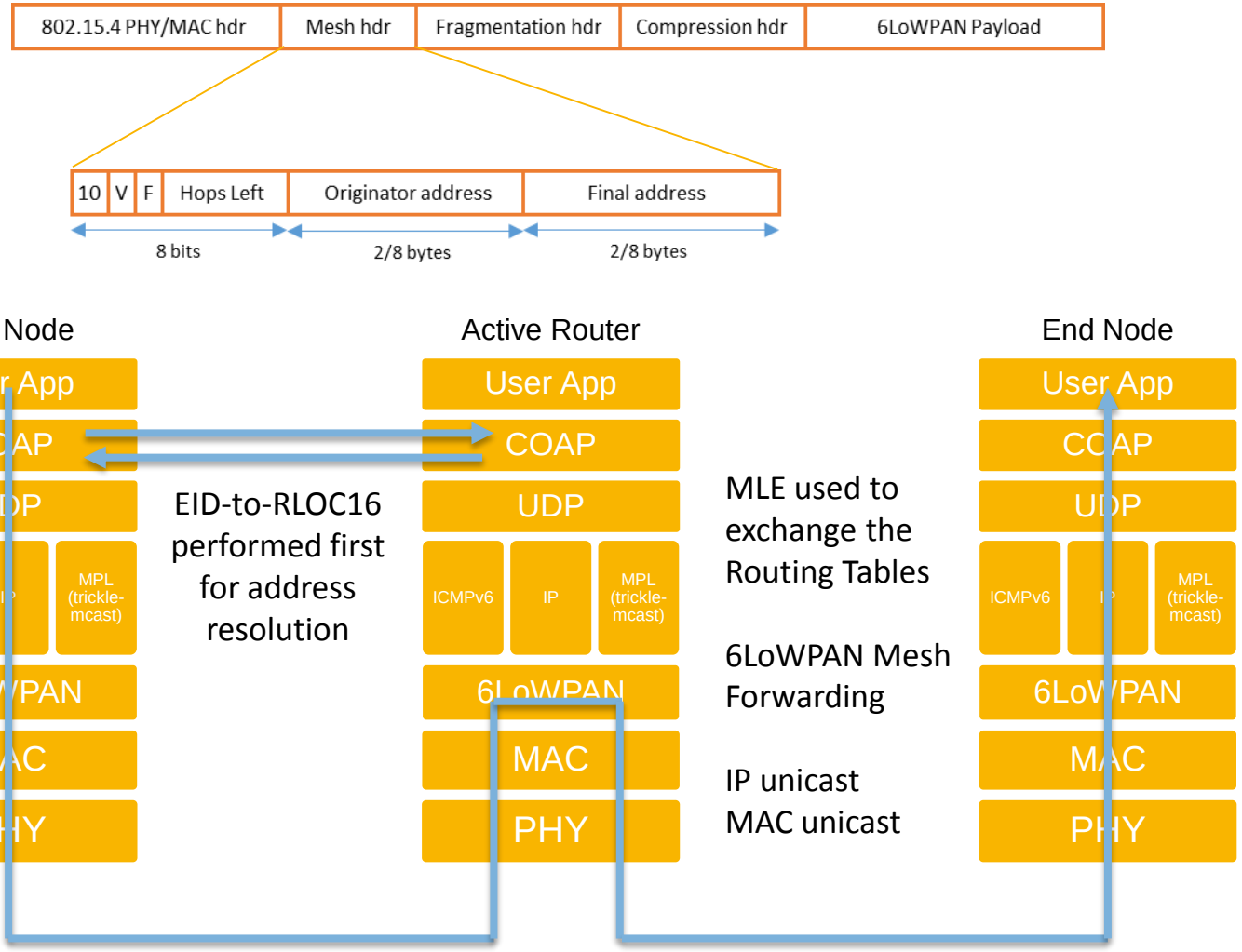


- Datagram size – The size of the original IPv6 packet.
- Datagram tag – A unique identifier of the IPv6 fragmented packet. It links all the fragments together.
- Datagram offset – Offset of the fragment relative to the original packet in multiples of 8 bytes.



Unicast Packets Forwarding – Using 6LoWPAN Mesh Header

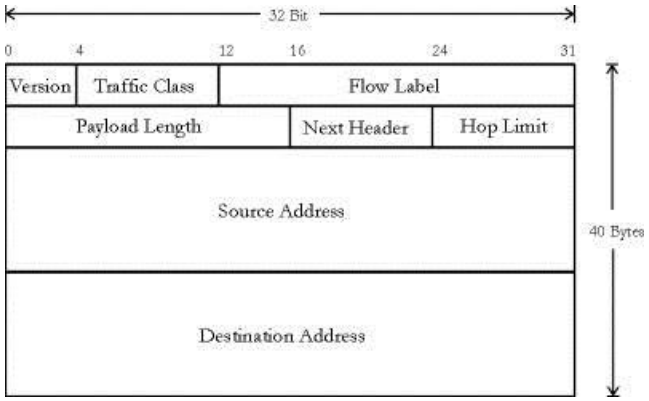
- Mesh Header is used for **multi-hop unicast forwarding**



IPv6 and ICMPv6

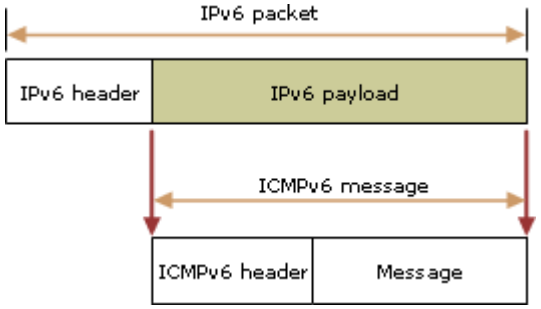
- IPv6 (RFC 2460)
  - provides an identification and location in the network via the IP address
  - 16 bytes addressing format (128 bits)
  - assures end-to-end datagram transmission across multiple IP networks
  - Fragments packets bigger than the MTU (maximum transmission unit)

IPv6 Header



- ICMPv6 (RFC 2463)
  - Internet Control Message Protocol version 6
  - Adds error handling to the IP protocol eg:
    - Destination unreachable (type 1) → no route, port unreachable, administratively prohibited, etc
    - Packet too big (type 2) → in case a non fragmented packet is being sent with a frame larger than the supported MTU
    - Echo request/reply (type 128 and 129) → ping functionality
    - etc…

ICMPv6 Message



Multicast Packet Forwarding – Using MPL module inside IPv6 Layer

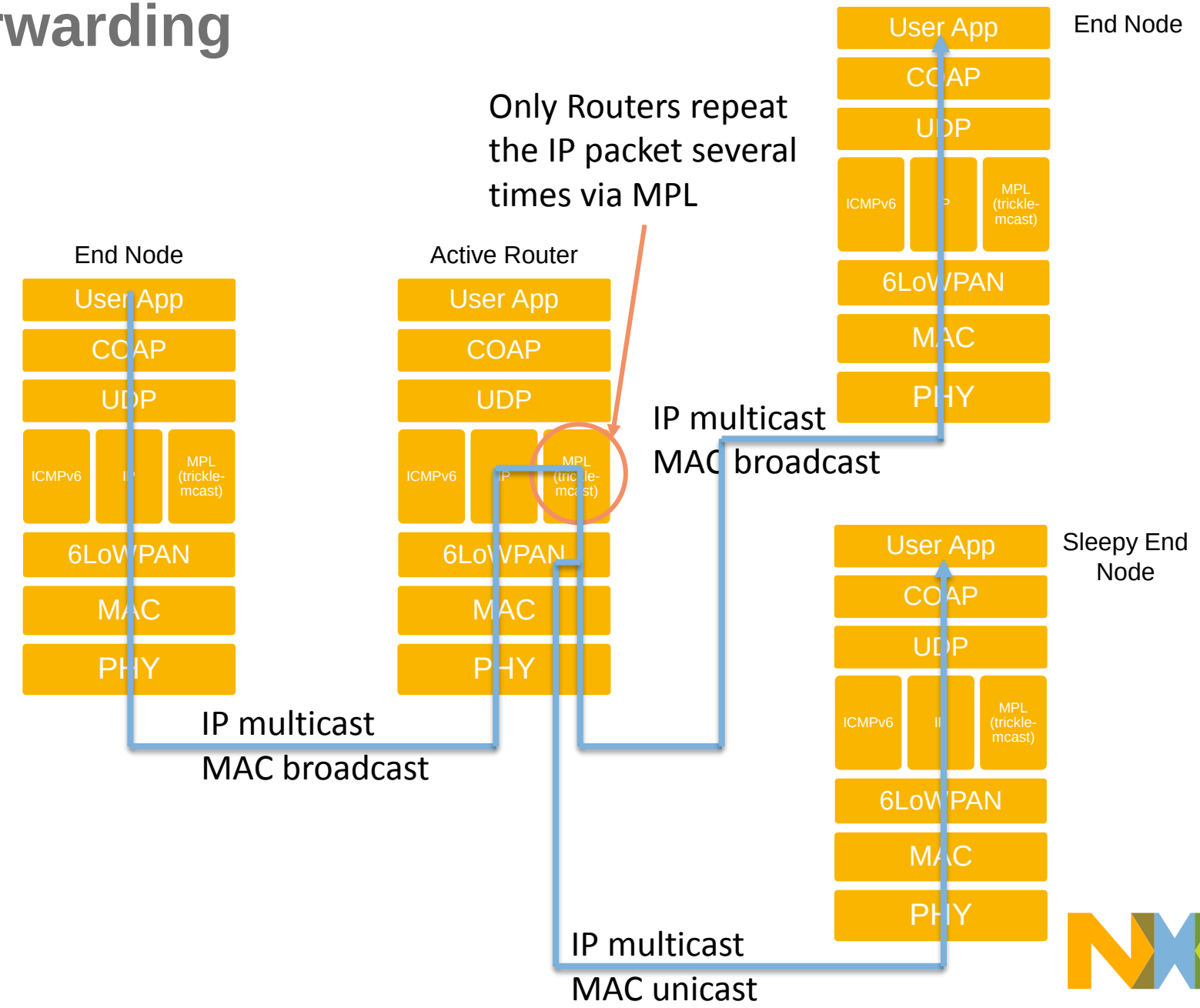
- The **MPL** protocol (Multicast Protocol for Low Power and Lossy Networks) is used for **multi-hop multicast messages forwarding** in the Thread network

- MPL allows retransmissions from Active Routers for more reliable and frame counters to prevent multiple instance of the messages to reach the application

- A **Trickle** timer algorithm allows for logarithmic slow-down of periodic multicast advertisements for propagating routing information and network data based on node density and freshness of information

Multicast Packets forwarding

- Different approach between normal powered devices and sleepy end devices:
  - Normal powered devices
    - Messages repeated by MPL
    - Broadcast at MAC layer
  - Sleepy end devices
    - Messages queued in the MAC layer of the parent router
    - Converted to unicast at MAC layer in the 6LoWPAN
    - Data retrieved by the end device via MAC-Poll commands

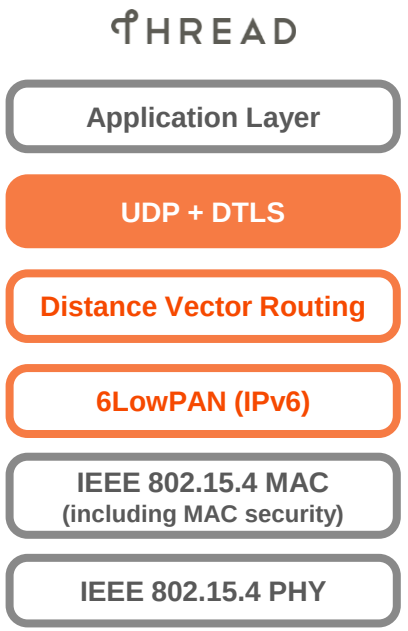


UDP – User Datagram Protocol
DTLS – Datagram Transport Layer Security

UDP is connectionless protocol - One program can send a load of packets to another with no handshake establishment. Suitable for applications that need **fast**, efficient transmission but the delivery of the packets is not guaranteed.

DTLS provides communication privacy (integrity, authentication and confidentiality) and other security properties such as replay prevention for datagram protocols

An additional crypto library like WolfSSL and PolarSSL is needed for using ciphersuites different than what is implemented in Thread (DTLS-jpake)



CoAP

- **CoAP** – Constrained Application Protocol in RFC 7252
  - Binary RESTful protocol, similar to a reduced HTTP
  - POST, GET, PUT, DELETE methods, ACK responses
  - CONfirmable (ACK requested) and NONconfirmable messages
  - No support today for CoRE features

- Thread uses CoAP for the majority of **multi-hop network management** and **commissioning** messages:
  - Request and Distribution of Router IDs from Leader
  - Propagation of external prefix from border routers
  - Resolving EID to RLOC address mapping and creating address caches
  - All Mesh Commissioning protocol application messages (with DTLS security)

- **Thread internal CoAP UDP ports**
  - 19789 for network management
  - 19779, 19782, 19786 for Commissioning (commissioner, relay, joiner)

- **Example applications CoAP UDP ports**
  - 5683 non secured CoAP
  - 5684 secured CoAP (CoAPs)

THREAD MANAGEMENT PLANE

Network Management

- Effective mesh networking using IEEE 802.15.4 requires identifying, configuring, and securing usable links to neighboring devices as the network's membership and physical environment change.

- **Management modules include:**
  - Mesh Link Establishment and IP address assignment
  - Network Device Roles (Leader, Eligible Router, Router, End Device) and parent selection
  - Network Routing, Data Propagation and Partitioning
  - Network Commissioning Protocol (Mesh COP)
  - Network Diagnostics (including network topology)

Use of MLE

- **MLE** – Mesh Link Establishment

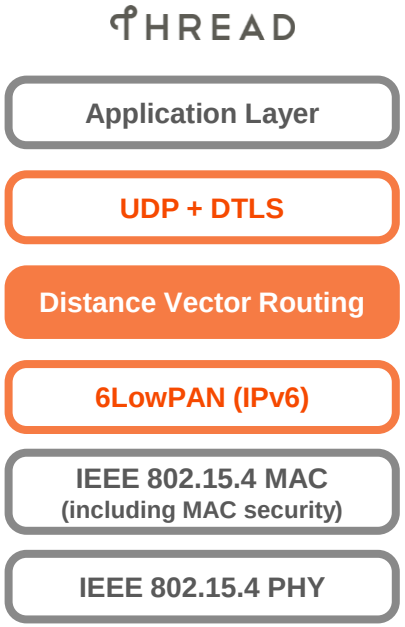
- Thread uses MLE for the majority of **single-hop** network messages to **negotiate links, share and propagate network information** between:
  - End Device Children and Parents
  - Neighboring Routers

- MLE is carried over UDP on port 19788

Network Routing

Similar algorithm to Routing Information Protocol next generation (RipNG):

- Distance Vector routing protocol
- All routers exchange with other routers their cost of routing in the Thread network in a compressed format using MLE (Mesh Link Establishment).
- Devices use IP routing to compute the routing table which is populated with a compressed form of a mesh unique local address for all routers and the appropriate next hop address.
- Routers inform their neighbors of topology changes periodically
- Packets forwarding is assured via 6LoWPAN at Link Layer



Routing Protocol

- All End Devices forward unicast data through their parent Active Router
- Active Routers advertise path cost and connectivity information to all other Active Routers and implement a distance vector routing algorithm
- All Active Routers keep a routing table based on cost advertisement from neighbors
- Routing Advertisements also include information on current Partition Leader and partition ID
- Routing Advertisements are single-hop MLE All Nodes multicast messages (FF02::01)

Routing Protocol – Link Margin, Quality and Cost

Link Margin	Quality	Link Cost
> 20dB	3	1
> 10dB	2	2
> 2dB	1	6
<= 2dB	0	infinite

The outgoing and incoming Link Qualities and Routing cost to an Active Router as advertised by any other Active Router can be encoded in 1 byte (per Router):

- Neighbor Out Quality – 2 bits (value 0 if self or not neighbor)
- Neighbor In Quality – 2 bits (value 0 if self or not neighbor)
- Routing Cost (minimal sum of Link costs) – 4 bits

Network Partitioning and Merging

- **Partitioning** – A set of Active Routers (with their children) which become disconnected from the current leader will create a new Thread network **partition**, having a new partition Leader.

- This can happen as current Leader has been turned off or removed or when nodes are moved out of connectivity range

- The new Leader chooses a new partition ID within the same Thread network

- If 2 or more different partitions become re-connected (Router or REEDs can hear routing advertisements from other partitions), the nodes in partitions with a lower partition ID will re-attach and **merge** to the partition with highest ID

Address Mapping, Caching and Duplicate Detection

- **EIDs are usually based on a random Interface ID (IID)**
- Thread nodes must resolve EIDs to RLOCs in order to route packets addressed to an EID
- CoAP network-wide multicast **Address Query** messages are used to find the RLOC mapped to an EID
- Parent Active Routers **cache** Address information for their children end devices
- **Duplicate Address Detection** is performed for EIDs – nodes advertise when detecting a duplicate and the EIDs are regenerated

REED Upgrade and Downgrade

- A **Router Eligible Device** joins the network as a **REED,** but will request a Router ID from Leader and **upgrade** to an Active Router if total number of Active Routers in partition less than a threshold (currently 16)
- If there are enough Active routers when joining, the node will remain a REED
- A REED will also request a Router ID when a new node attempts to join and there are not enough Active Routers in the range of the new node to accept that as a child
- An Active Router will release the Router ID and **downgrade** to a REED when the total number of Active Routers is above a threshold (currently 23) and other inter-connectivity criteria for neighbors and children are met

Network Data

- **Network Data** – Set of TLVs used by the Leader to propagate information about:
  - Active Servers or Border Routers advertising external global prefix information or services
  - Information about active Commissioners

- **Network Data can be:**
  - Stable: expected to have a long lifetime, propagated in a format accessible to all nodes, including sleepy End Devices
  - Temporary: expected to have a relatively shorter lifetime, propagated to Router/REEDs

- Network Data is managed through MLE Advertisements

Management Diagnostics

- Can be retrieved via CoAP on port 19789
- Example: coap://[fd01::3ead:d0e0:6e62:cd9a:46ff]:19789/d/dg

TLV Value	Name	Format	Can Reset?
0	Source Address (EUI-64)	8-byte address	N
1	Address16 (16-bit)	2-byte address	N
2	Mode (Capability information)	Same format at Mode TLV (Type 1 of MLE)	N
3	Timeout (Sleepy polling rate)	2-byte value	N
4	Connectivity	Same as the Connectivity TLV (MLE TLV Type 15)	N
5	Routing Table	Same as the Route64 TLV (MLE Type 9)	N
6	Leader Data	Same as TLV Type 11	N

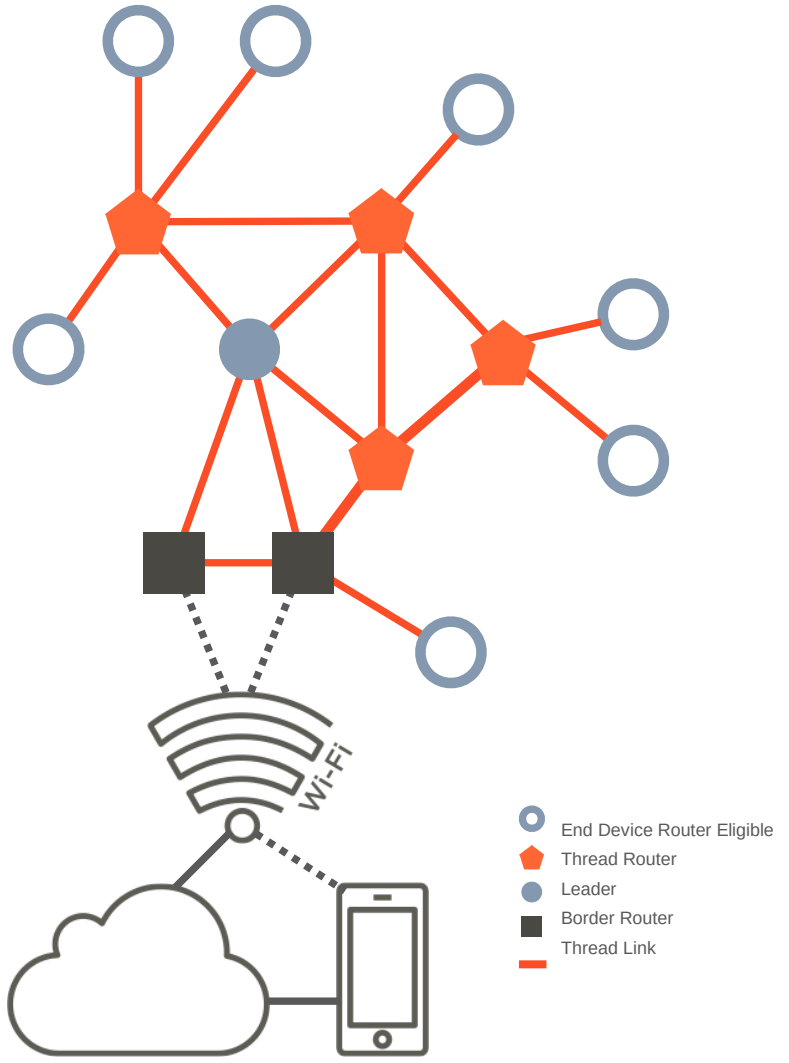
Management Diagnostics - Continued

TLV Value	Name	Format	Can Reset?
7	Network Data (Border Routers)	Same as TLV Type 12	N
8	IPv6 address list	List of all IPv6 addresses registered by the device	N
9	Packets sent	2-btye value	Y
10	Packets received	2-byte value	Y
11	Packets dropped on transmit	2-byte value	Y
12	Packets dropped on receive	2-byte value	Y
13	Security errors	2-byte value	Y
14	Number of retries	2-byte value	Y
15	Voltage	2-byte value [mV]	N
16	Child Table	Structure containing information on all children	N

THREAD SECURITY

Security Overview

- Simple Secure Commissioning
  - **User authorizes devices onto the network using smart phone**
  - **GUI rich device within network (e.g.: Thermostat) can also be used to authorize devices**
- Security session established between new device and commissioning device to authenticate and provide credentials – **DTLS** with **Elliptic Curve J-PAKE** are used for commissioning
- Once commissioning session is done – device attaches to the Thread mesh network
- MAC and Network Mesh Management layers use **AES128 encryption** for **all messages** as defined by **IEEE 802.15.4** specification
- **Automatic key rotation** - Thread networks provide mechanisms for parameterizing, changing, and negotiating shared network keys change after a time interval
- As a generic IP layer network specification, Thread can carry multiple **DTLS** or **TLS** flavors at the application layer in a flexible manner



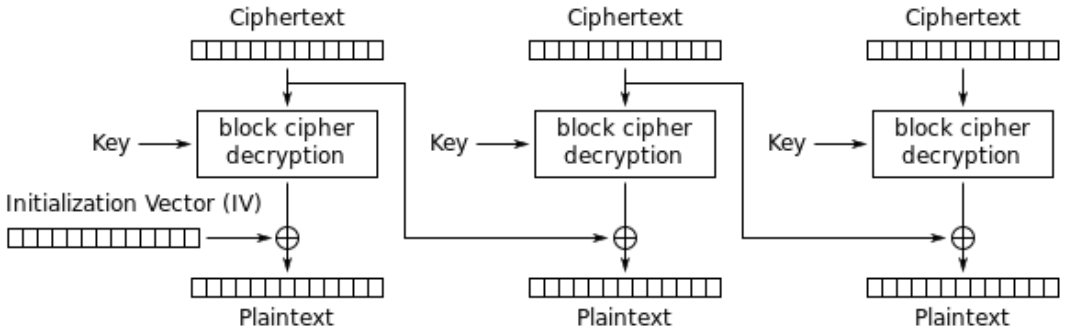
IEEE® 802.15.4 MAC Security

- The cryptographic mechanism in this standard is based on symmetric-key cryptography and uses keys that are provided by higher layer processes.

- The cryptographic mechanism provides particular combinations of the following security services:
  - — *Data confidentiality: Assurance that transmitted information is only disclosed to parties for which it* is intended.
  - — *Data authenticity: Assurance of the source of transmitted information (and, hereby, that information* was not modified in transit).
  - — *Replay protection: Assurance that duplicate information is detected.*

- The security module uses a generic combined encryption an authentication block cipher mode(**CCM***).

- The block cipher used in this standard shall be the advanced encryption standard **(AES)-128**, as specified in FIPS Pub 197.



Cipher Block Chaining (CBC) mode decryption

Elliptic Curve J-PAKE

- Thread uses **Elliptic Curve** variant of **J-PAKE**, with the NIST P-256 curve
- J-PAKE is a Password Authenticated Key Exchange (PAKE) protocol “juggling” public keys in a verifiable way (hence the “J”)
- Uses a variant of Elliptic Curve Diffie-Hellmann to provide **key agreement**
- Supplements Diffie-Helmann with Schnorr signatures as a non-interactive zero-knowledge (NIZK) proof mechanism to **mutually authenticate two peers and to establish a shared secret between them based on a passphrase**
- A (D)TLS 1.2 [RFC 6347] handshake is used for transporting the key exchange

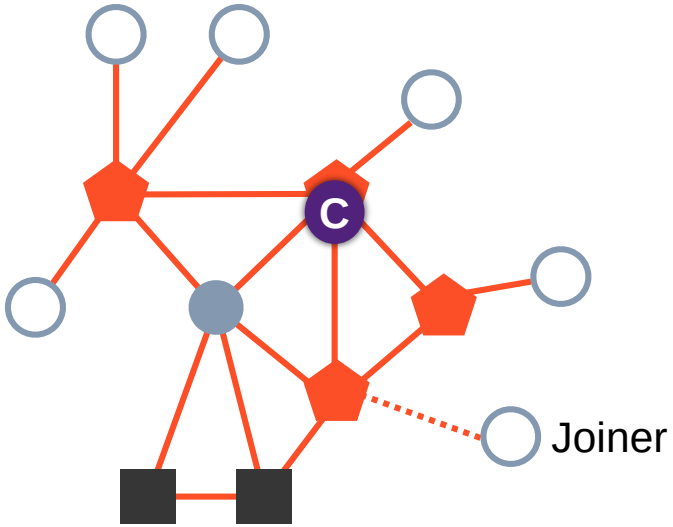
J-PAKE Characteristics

- **Off-line dictionary attack resistance** - It does not leak any password verification information to a passive/active attacker
- **On-line dictionary attack resistance** - It limits an active attacker to test only one password per protocol execution
- **Forward secrecy** - It produces session keys that remain secure even when the password is later disclosed
- **Known-key security** - It prevents a disclosed session key from affecting the security of other sessions
- Since 2015, J-PAKE has a formal security proof: http://www.normalesup.org/~fbenhamo/files/publications/SP_AbdBenMac15.pdf

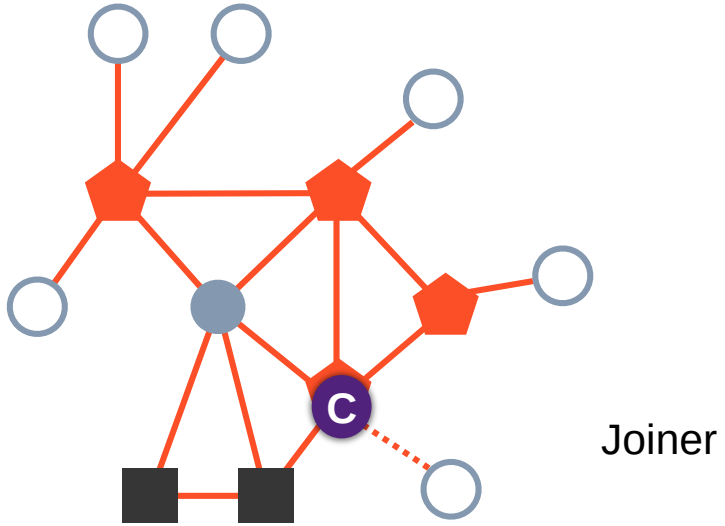
THREAD COMMISSIONING

Topology in-band (in-mesh) commissioner

Expanded



Collapsed

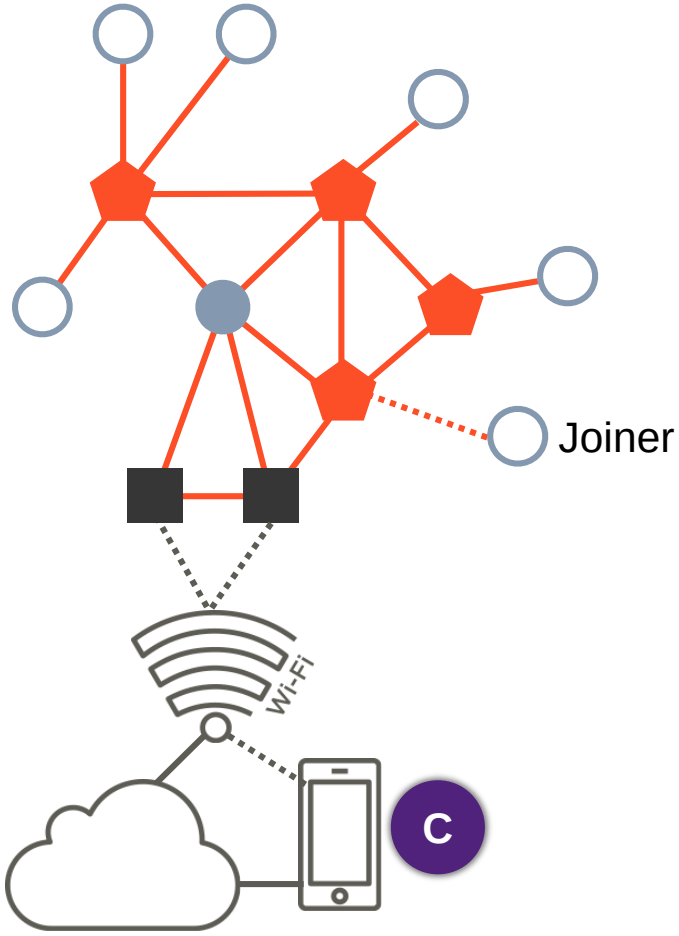


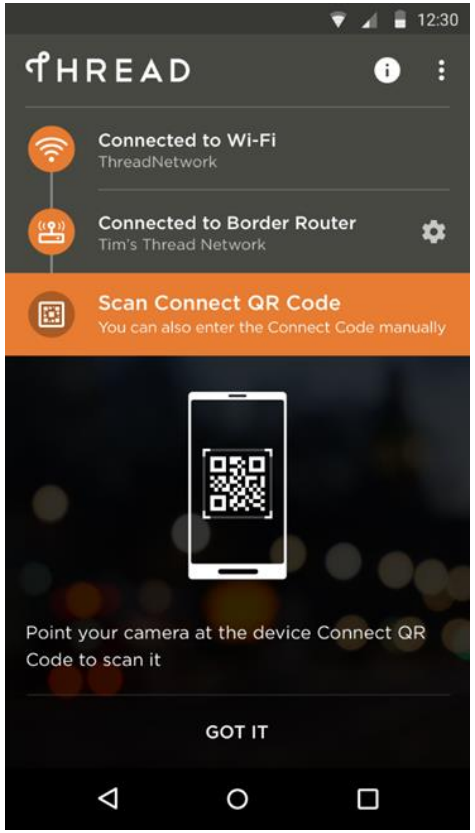
Joiner

C Commissioner

Topology out-of-band commissioner (eg. WiFi, BLE, NFC, other)

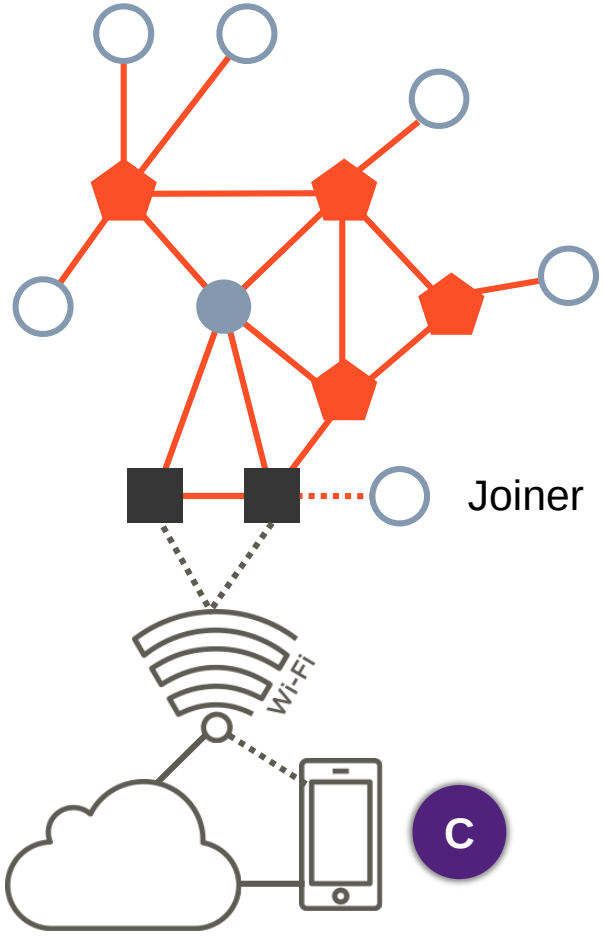
Expanded





Thread Commissioning Mobile App

Collapsed



Joiner

C Commissioner

Commissioner Mobile Application

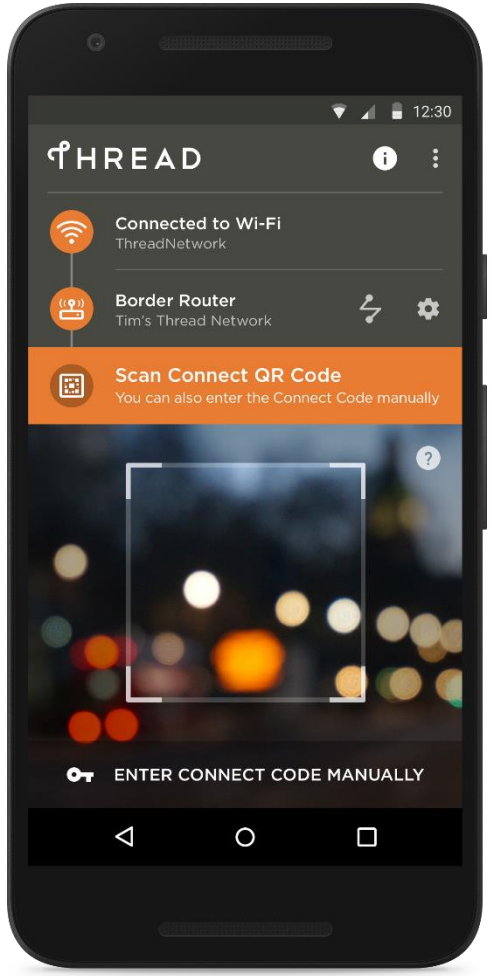
Simple, consumer friendly method for adding devices onto a Thread network

App uses QR Code or simple user friendly key to identify joining product

Mobile device attaches to Thread Network through Border Router to add device to network

Thread group developed a sample commissioning app that is available to Thread Sponsor and Contributor members

Available in iOS and Android



Example: Thread Device Out-of-Band commissioning with BLE

Smart Door Lock contains KW40Z + Host MCU / KW41Z Multi-Protocol Radio running

- Bluetooth Low Energy
- 802.15.4 MAC/PHY

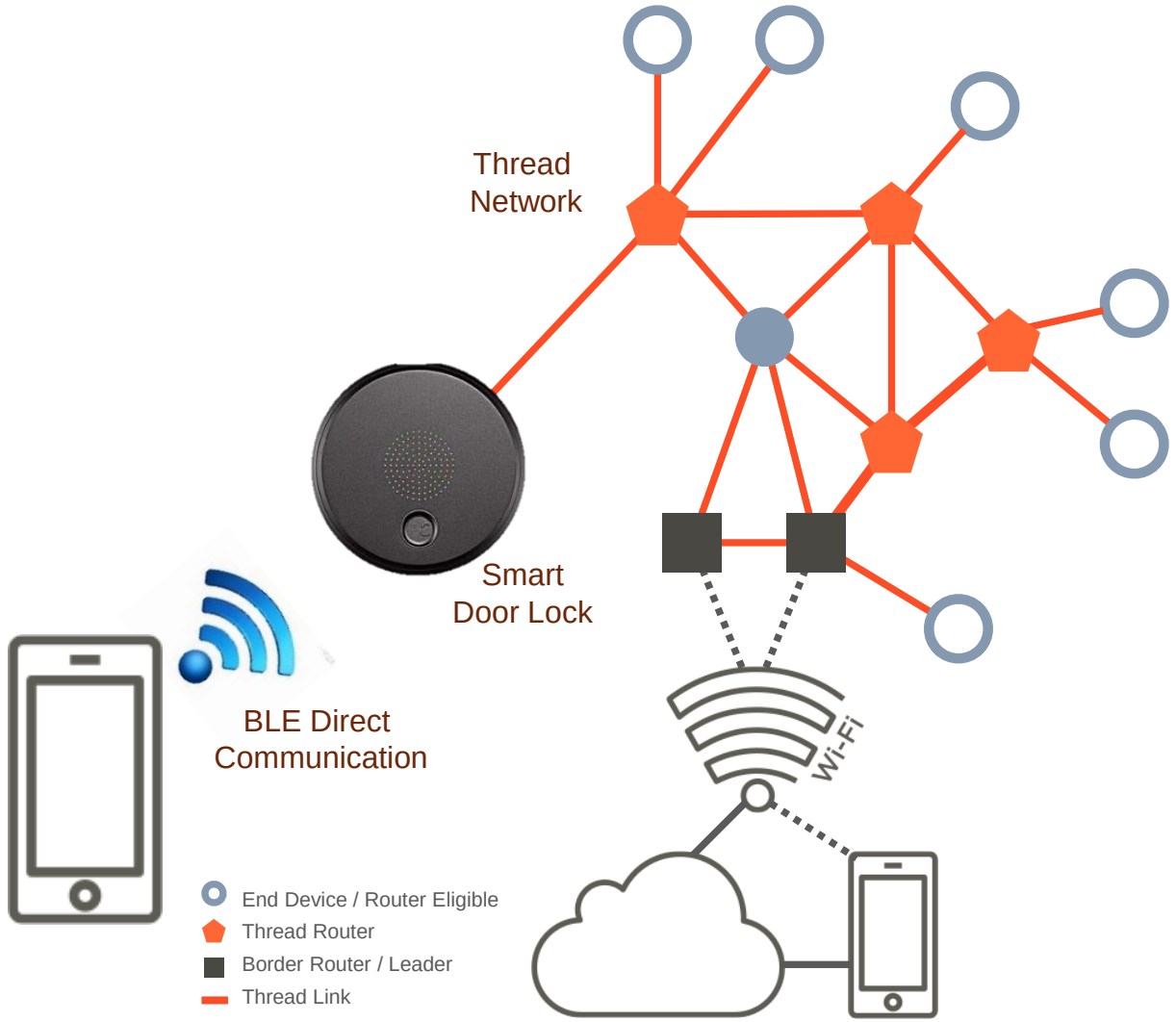
Host MCU / KW41Z runs Thread Stack

Local and Remote Control

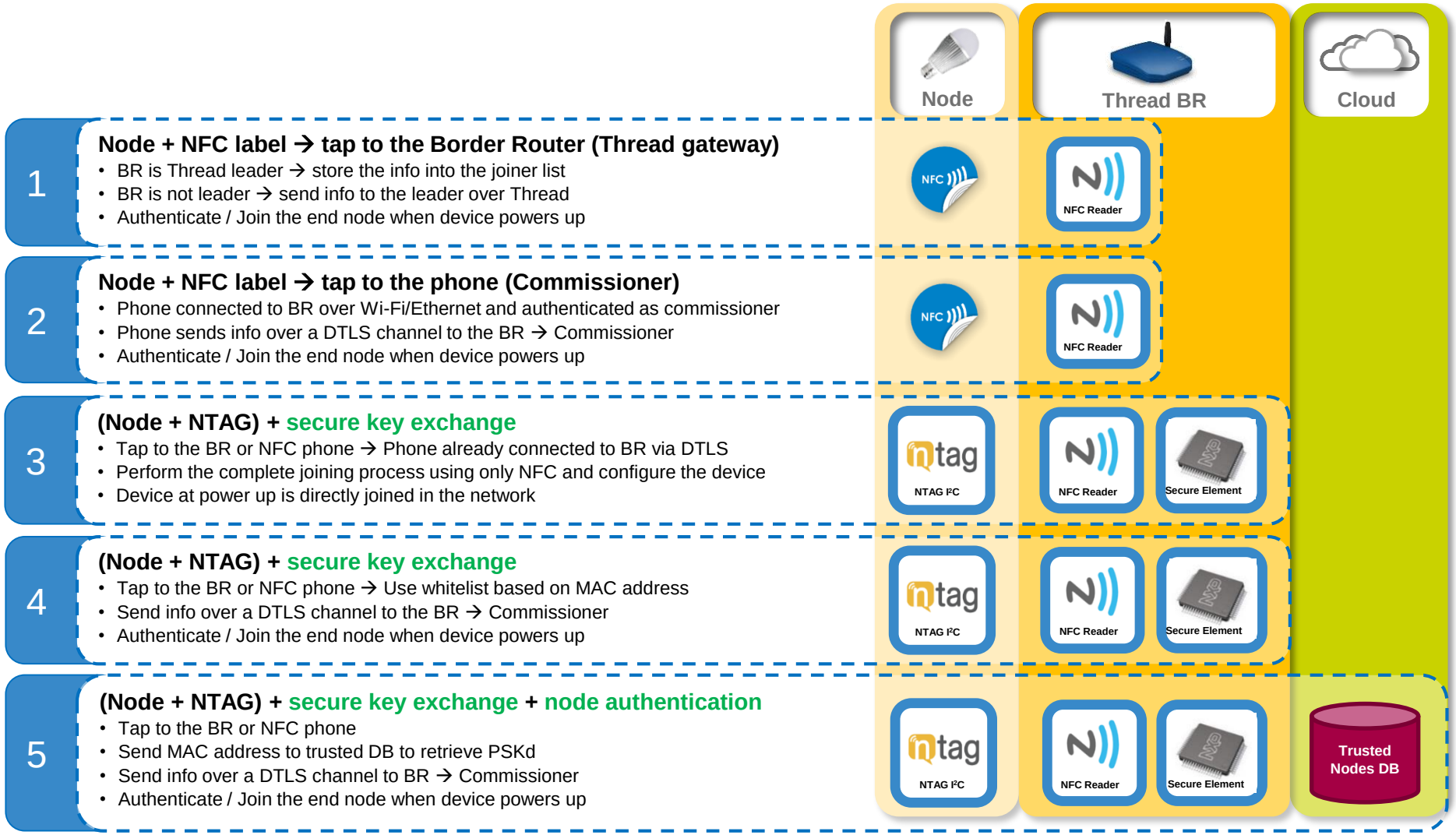
- Control locally from BLE enabled smartphone/tablet
- Control remotely using cloud connected Thread mesh network

Thread management via BLE

- Out of band commission and decommission devices to / from the Thread network
- Perform device management and debugging over BLE



Example: Thread Device Out-of-Band commissioning with NFC



COMMISSIONER

Commissioner

Protocol

Discovery Commissioner Candidate (smartphone with WiFi) discovers a Thread Network thorugh one of its Border Routers

Authentication Commissioner Candidate securely connects to the Thread Network using the commissioning credential

Registration Commissioner Candidate registers its identity with its Border Router

Thread Management

Petitioning border router unicast to the Thread Network Leader a request to petition its Commissioner Candidate to be one elected Commissioner

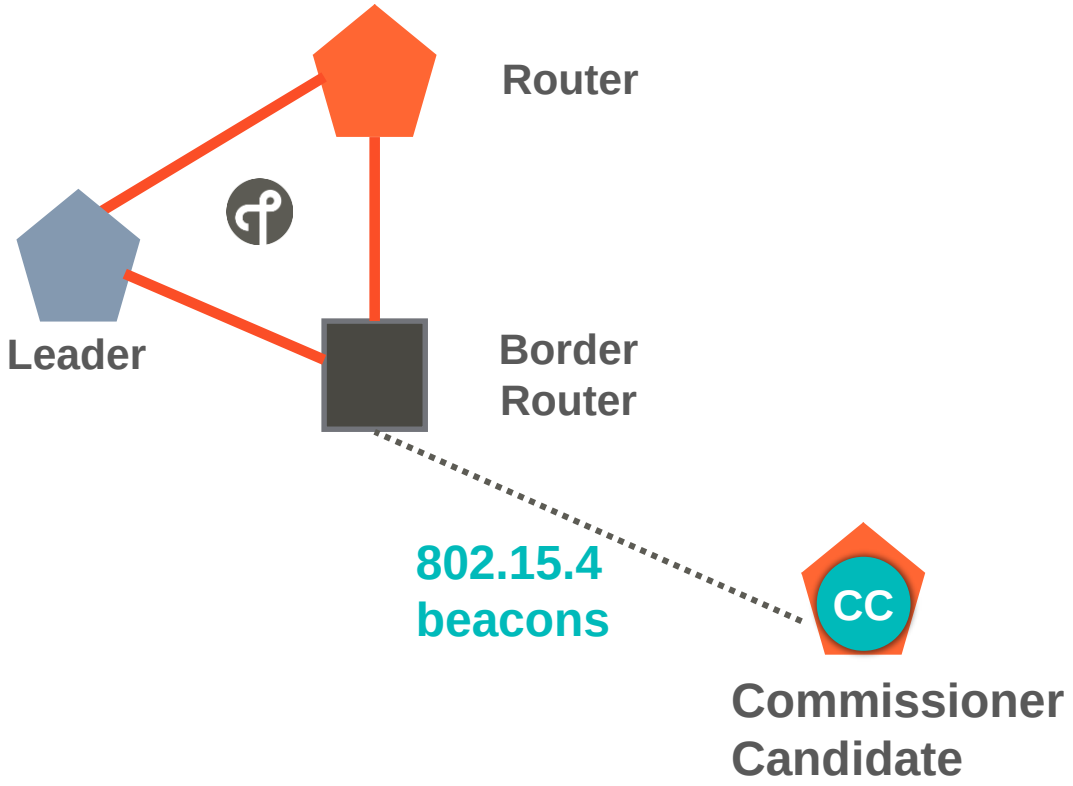
Management commissioners may manage the network by getting and setting parameters such as: commissioner credentials, network name, security policy.

Commissioner Protocol

Discovery Native 802.15.4

Native Commissioners use the discovery process used by Joiners using 802.15.4 beacons

Commissioner port contained in the beacon, or default to a fixed port



Commissioner Protocol

Discovery Ethernet / WiFi

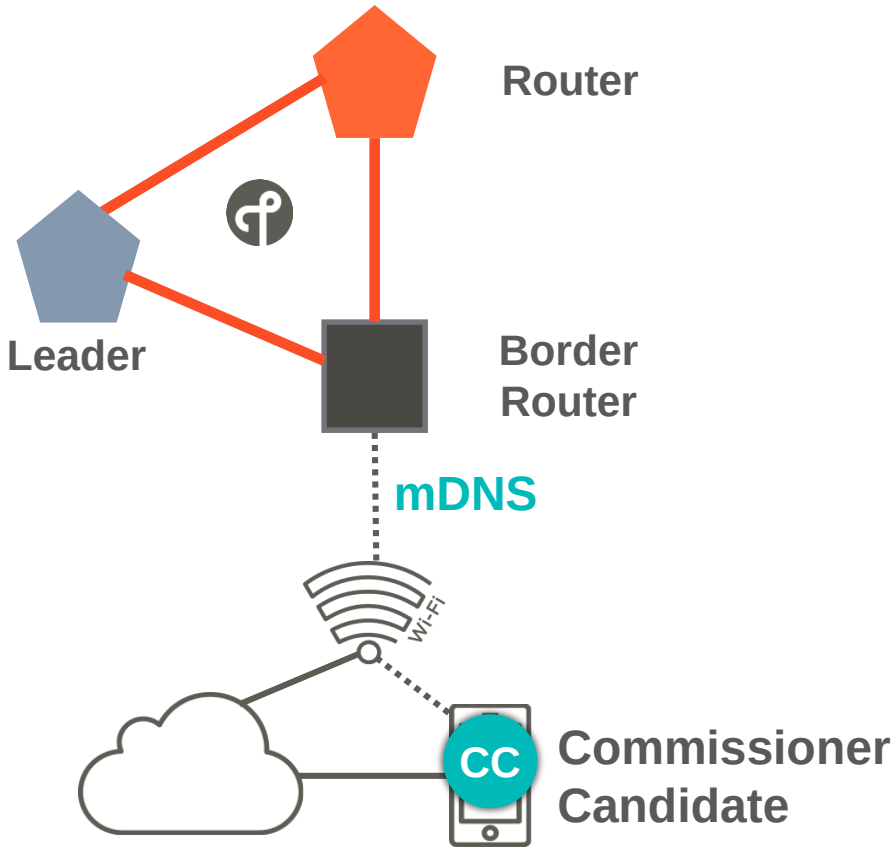
Border Router advertise the Thread Commissioning service using mDNS-SD _thread-net._udp.local.

Commissioner Candidate looksup for

→Network name

→Commissioner Port

→ Network XPANID



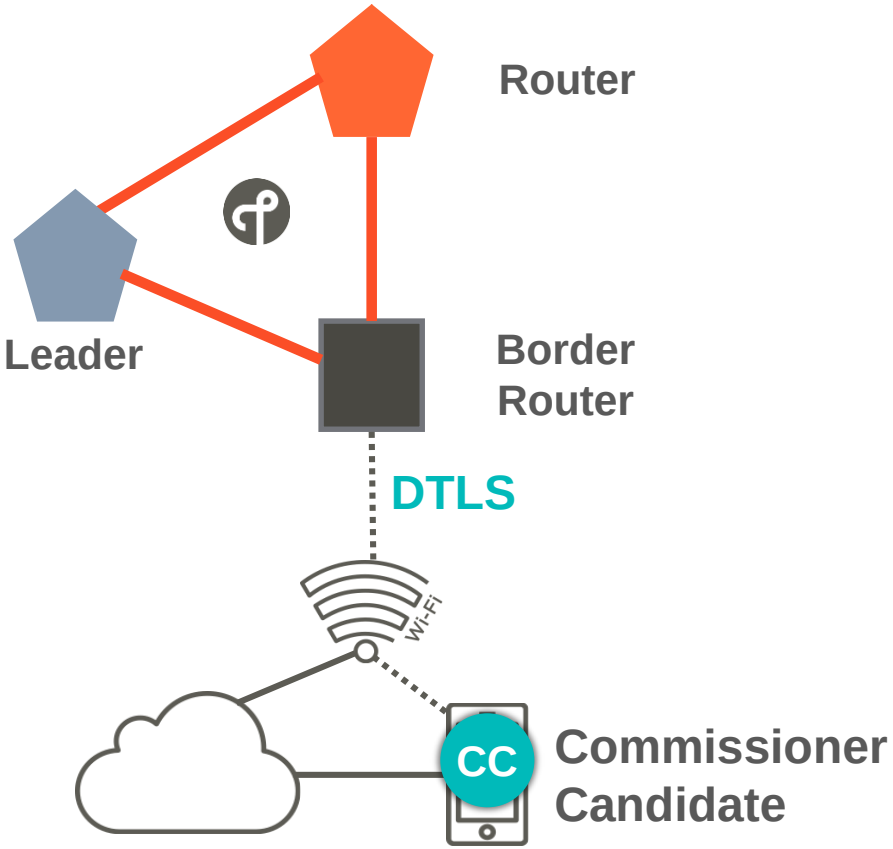
Commissioner Protocol

Authentication

Establish a secured client / server socket connection between the CC and the Border Router via DTLS known as **Commissioning Sesion**

Uses the UDP port advertised during the Discovery phase known as the **Commissioner Port** MC default to19779

PSKc is the credential used to stablish a Commissioning session



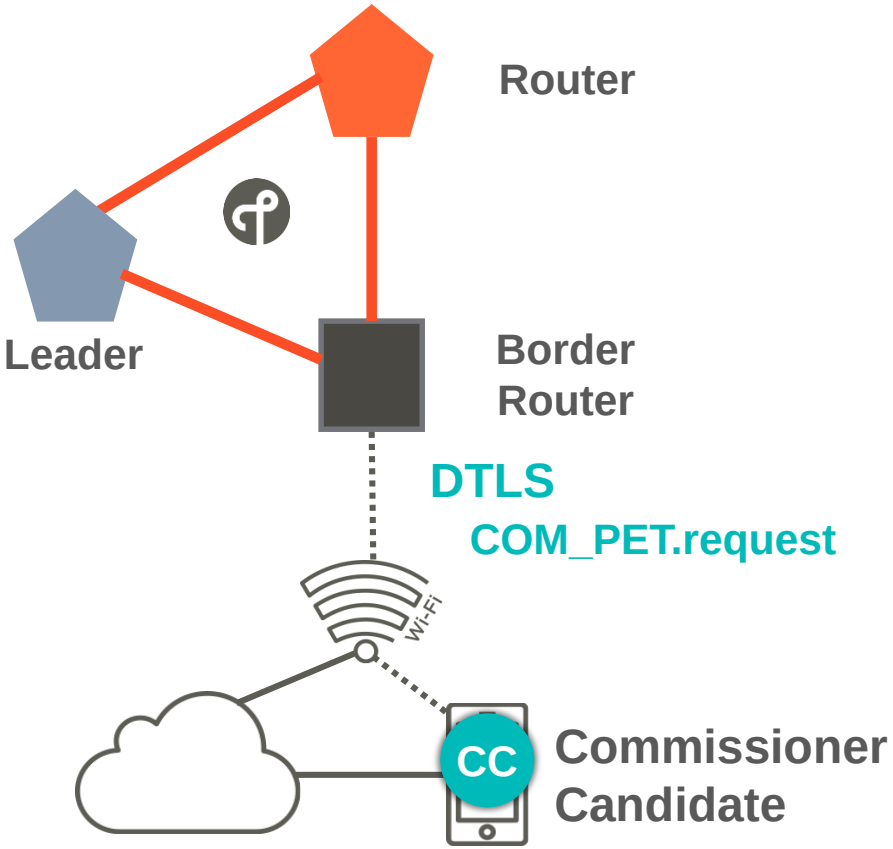
Commissioning Session

Commissioner Protocol

Registration

Commissioner Candidate registers its identity with the Border Router by sending the COMM_PET.request message

Failed attempts by a Commissioner to establish its authority shall-be rate-limited



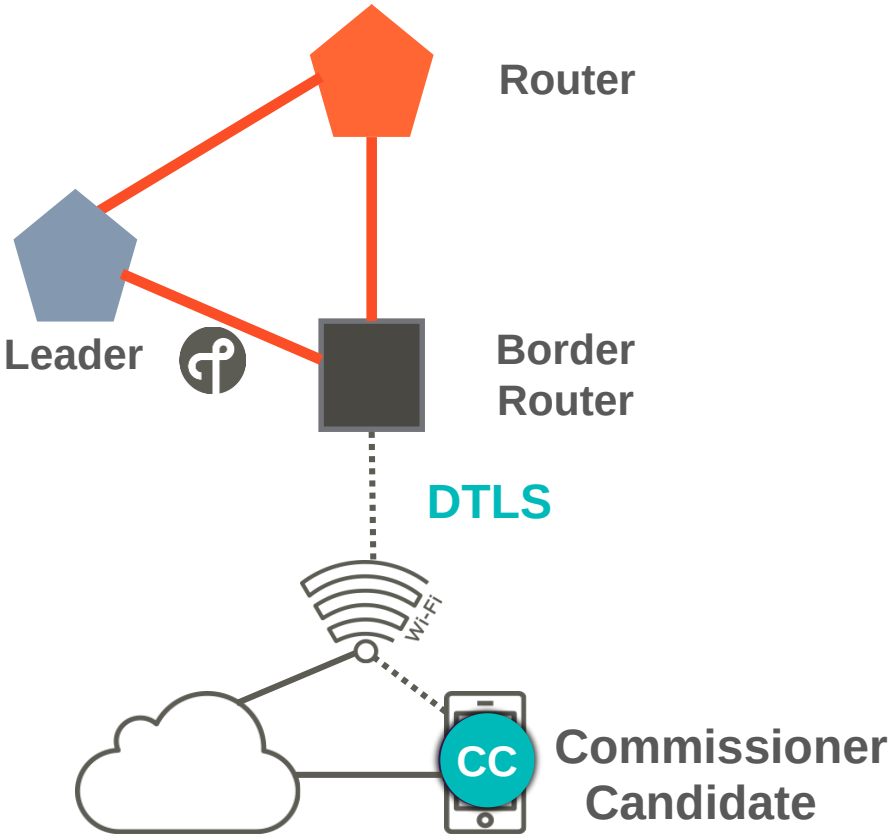
Commissioning Session

Thread Management

Commissioner Petitioning

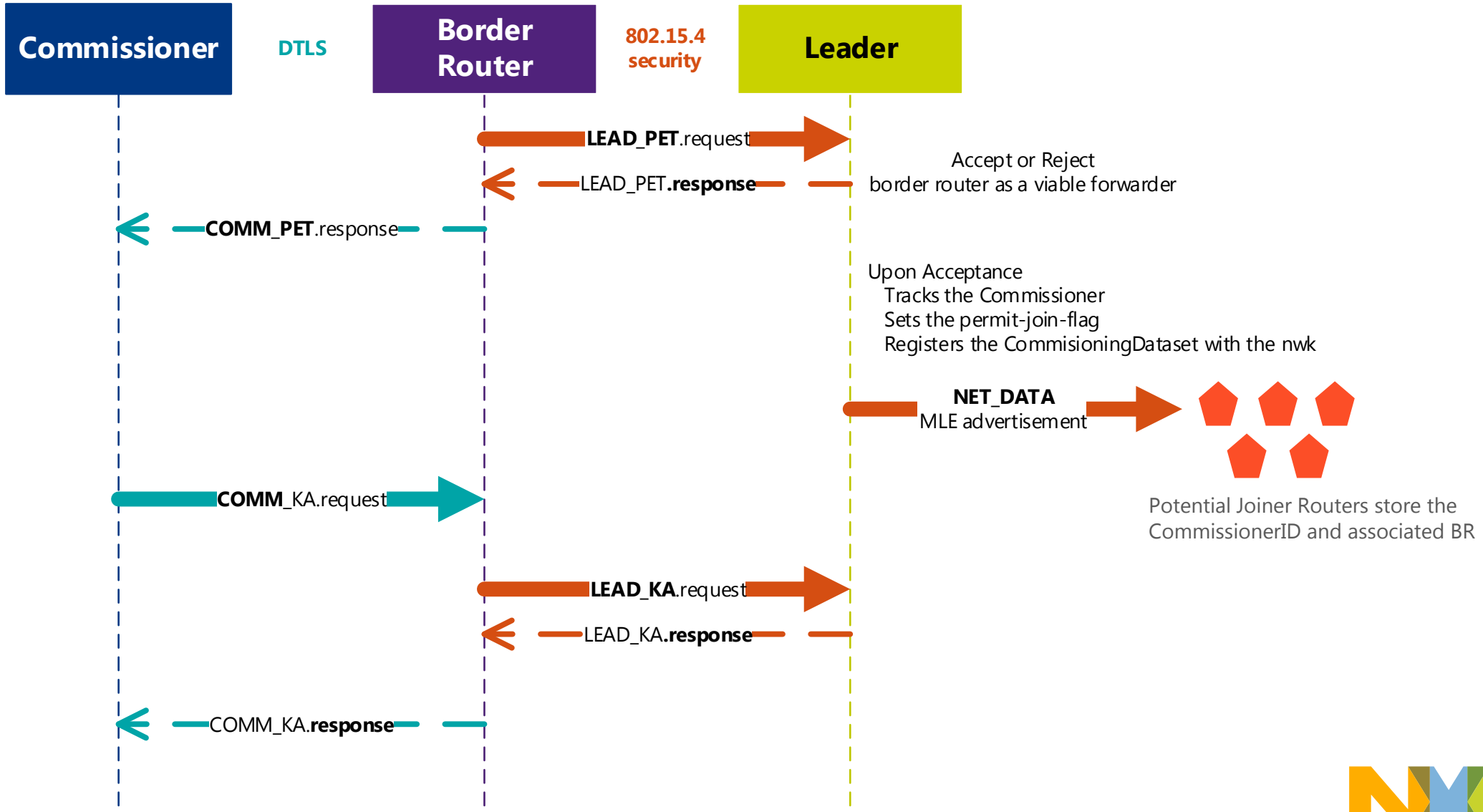
BR unicast to the Leader a request to petition its Commissioner Candidate to become the one elected Commissioner

Leader responds to BR Accept or Reject. In case for an Accept the Commissioner Candidate becomes the Commissioner.



Commissioning Session

Thread Management – Commissioner Petitioning



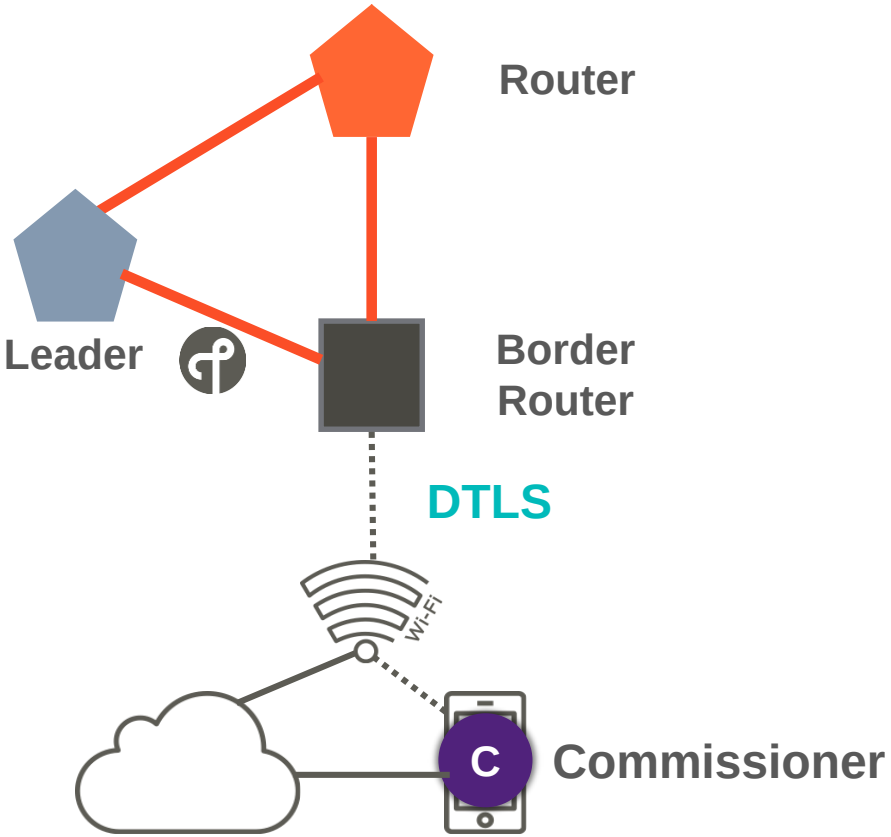
Thread Management

Commissioner Managment

Commissioner may manage the network parameters such as

- Commissioning Credential
- Network Name
- Security Poly

Constructs steering data that signals to expected joiners.



Commissioning Session

JOINER

Joiner

Joiner Protocol

Discovery Joiner discovers the Thread Network using 802.15.4 active scan by sending a beacon request

Provisional Join unsecure local-only link to the joiner router

Joiner Authentication DTLS handshake messages to a Joiner Router

Joiner Finalization

Entrust handoff of network credentials to the Joiner

Provisioning if the joiner appealed for a specific commissioning application, do vendor-specific provisioning

Session Close DTLS Alert mechanism.

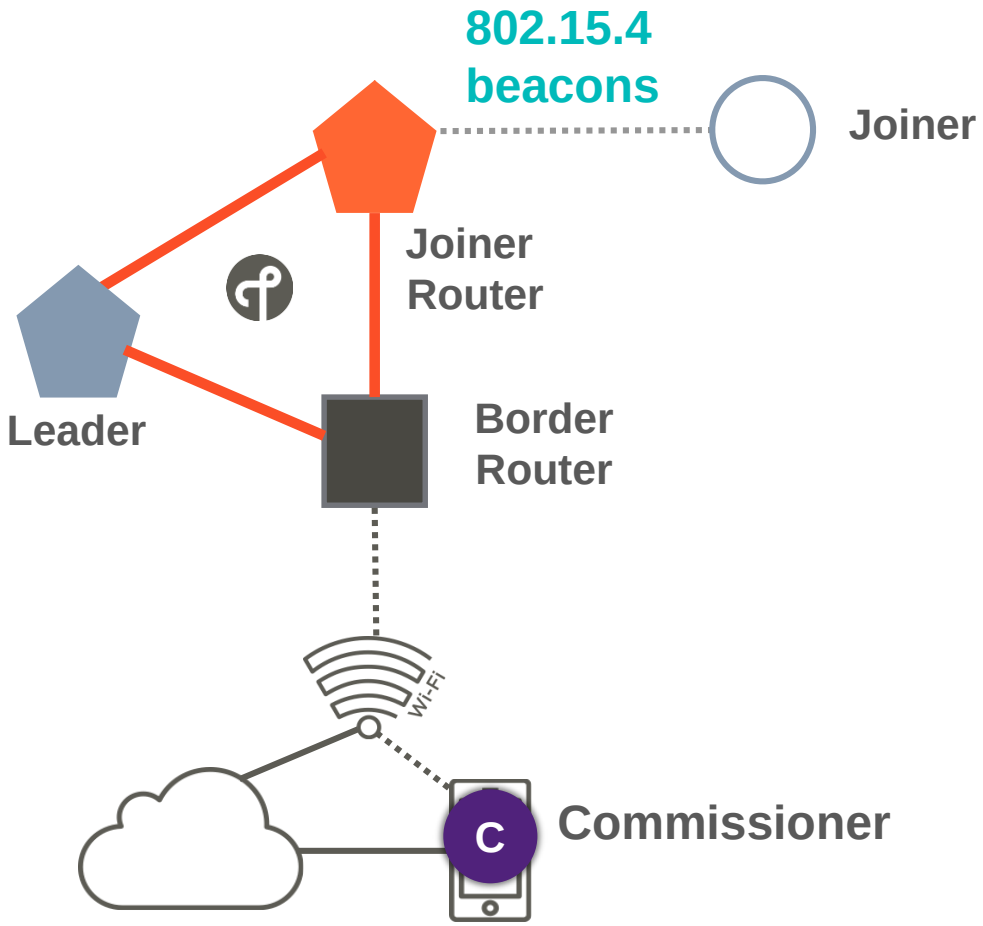
Joiner Protocol

Discovery

Joiner sends a standard 802.15.4 active scan by sendind a beacon request on every channel

The beacon from the Joiner Router carries steering data in the payload like

- Network Name
- Extended PANID
- Steering Data TLV (optional)

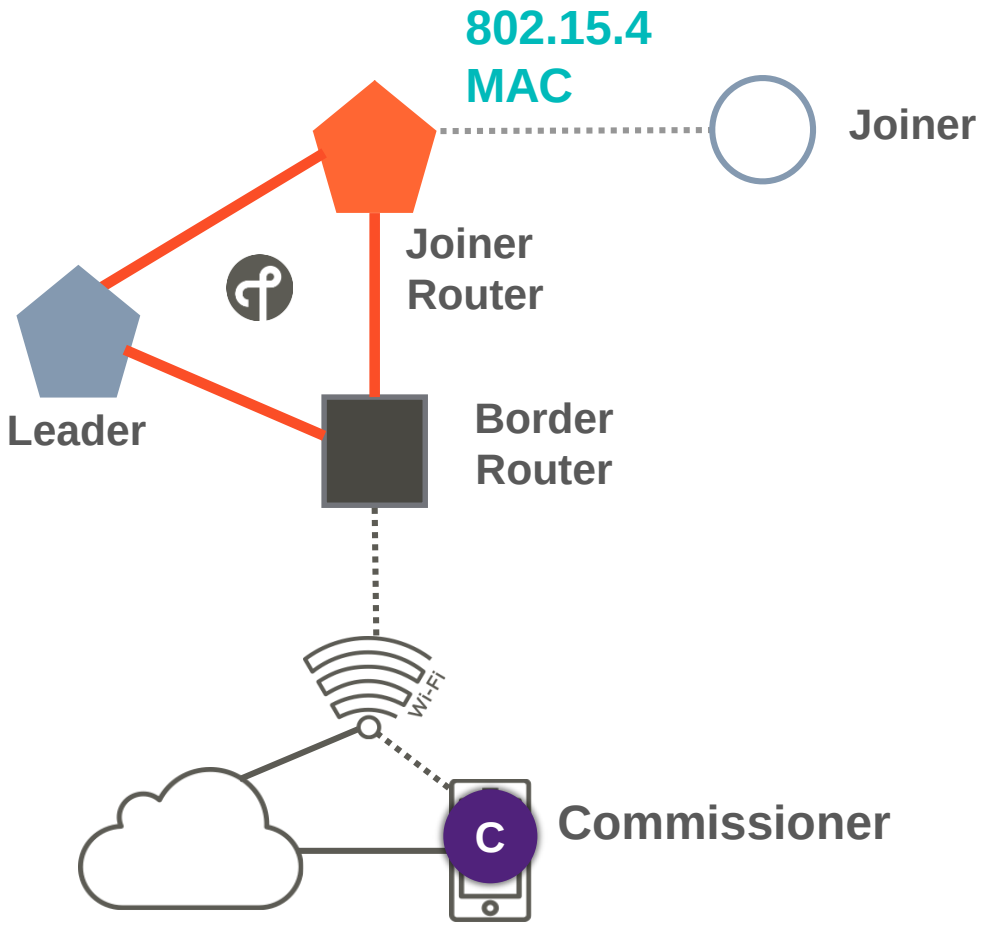


Joiner Protocol

Provisional Join

Joiner stablishes an unsecured local-link to the Joiner router

Joiner configures its MAC layer with the network parameters (Channel, xPANID, Network Name)

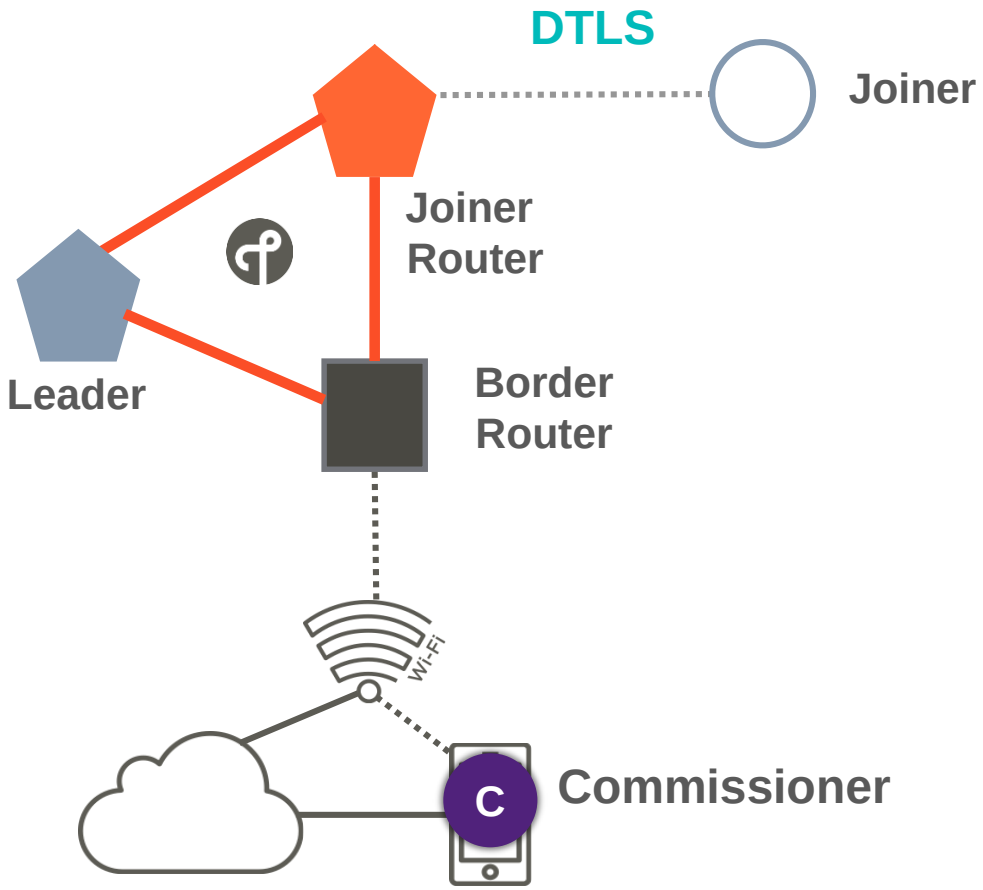


Joiner Protocol

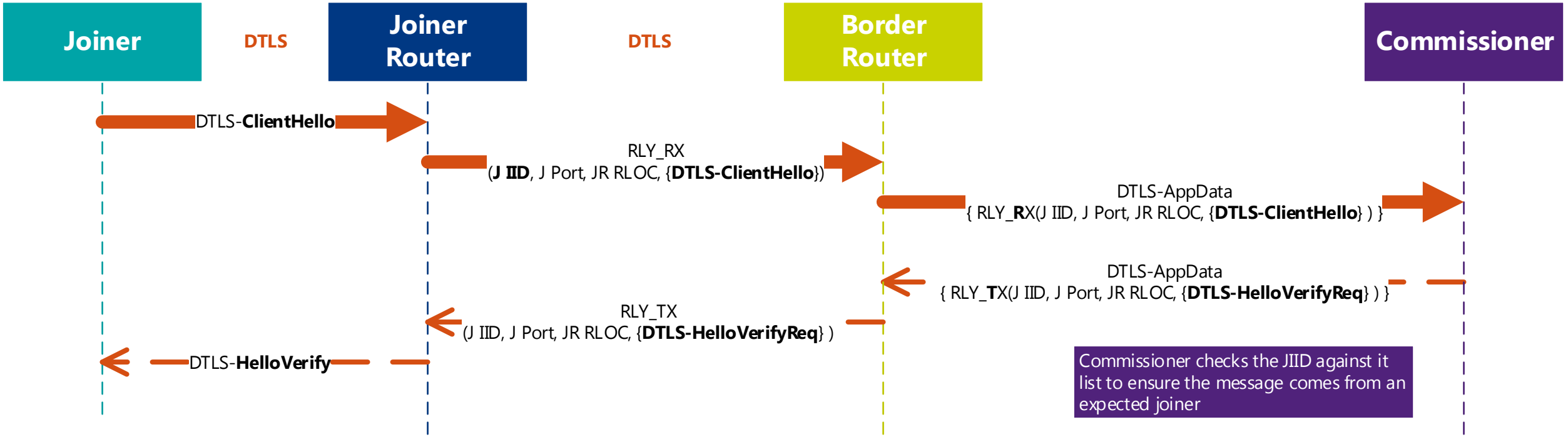
Authentication

Joiner sends raw DTLS handshake messages transported over UDP to a Joiner Router

The Border Router forwards to the Commissioner the DTLS messages received from the Joiner Router



Joiner Protocol – Authentication



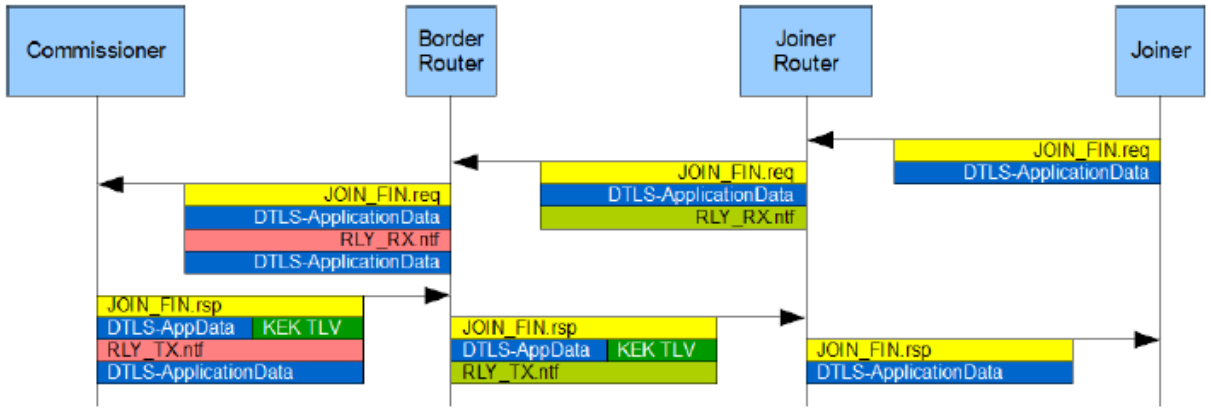
JIID → Derived from the EUI64 of the joiner & source UDP port

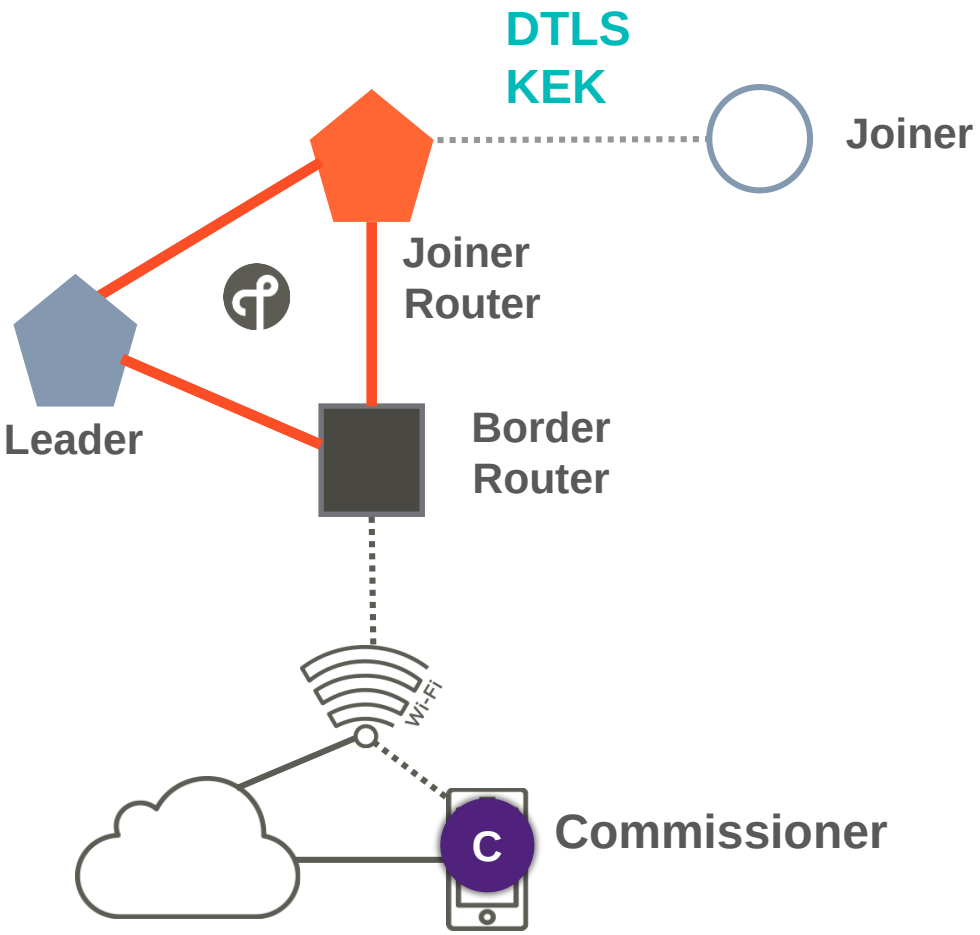
Joiner Finalization

Entrust

Process to handoff Network Credentials to the Joiner (Network Master Key, Network Mesh-Local ULA)

The Commissioner uses KEK (Key Encryption Key) to share a secret to the Joiner throught the Joiner Router



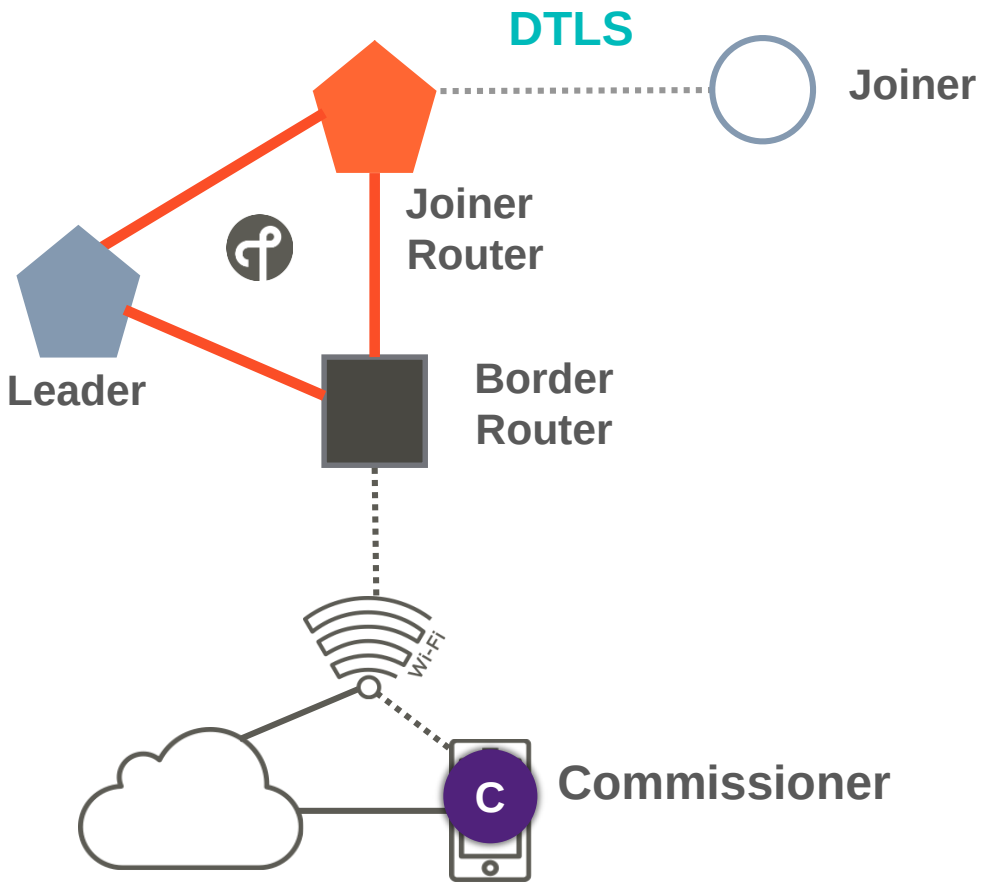


Joiner Finalization

Provisioning

If the joiner appealed for a specific commissioning application, the joiner session remains open for vendor-specific commissioning. The joiner session will close when the vendor-specific protocol specifies it.

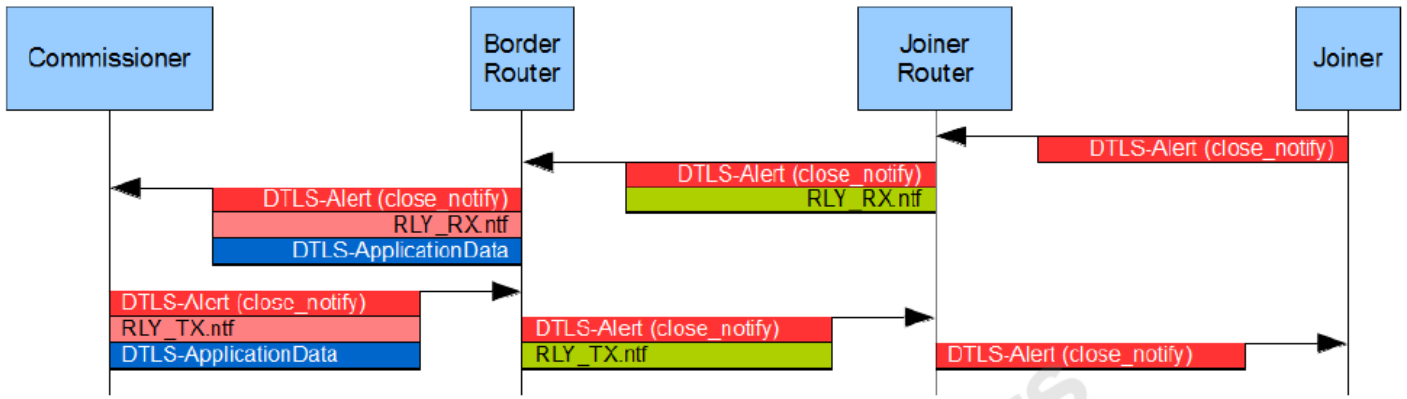
In all other cases the joiner session is closed inmmediatly after Joiner Entrust

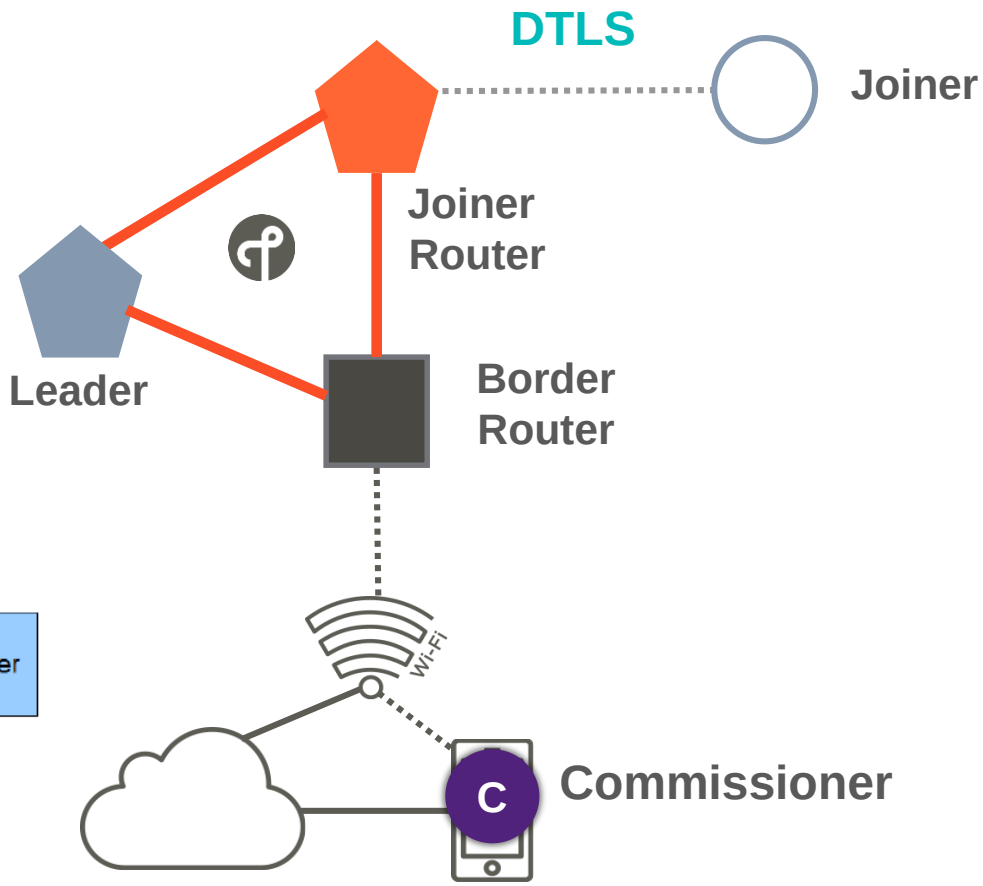


Joiner Finalization

Joiner Session Close

When the joiner session is closed, it MUST be closed using the standard DTLS-Alert round trip message exchange





Joiner Sequence – Wireshark Decoding

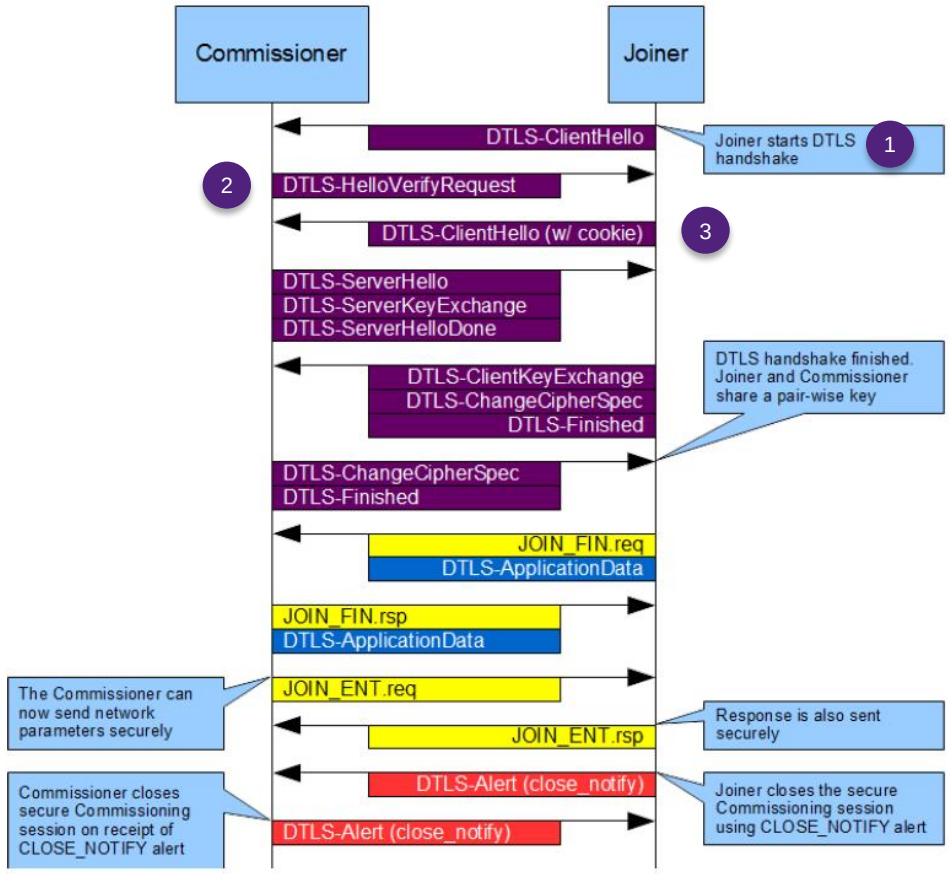


Figure 17. Joiner–Joiner Router/Commissioner Sequence

```
11 Client Hello                                                          DTLSv1.2
12 Ack                                                                   IEEE 802.15.4
13 Hello Verify Request                                                  DTLSv1.2
14 Ack                                                                   IEEE 802.15.4
15 Data, Dst: 4e:db:a9:f9:08:68:7f:a2, Src: 4b:54:97:dd:5a:71:f3:fa      6LoWPAN
16 Ack                                                                   IEEE 802.15.4
17 Data, Dst: 4e:db:a9:f9:08:68:7f:a2, Src: 4b:54:97:dd:5a:71:f3:fa      6LoWPAN
18 Ack                                                                   IEEE 802.15.4
19 Data, Dst: 4e:db:a9:f9:08:68:7f:a2, Src: 4b:54:97:dd:5a:71:f3:fa      6LoWPAN
20 Ack                                                                   IEEE 802.15.4
21 Data, Dst: 4e:db:a9:f9:08:68:7f:a2, Src: 4b:54:97:dd:5a:71:f3:fa      6LoWPAN
22 Ack                                                                   IEEE 802.15.4
23 Client Hello                                                          DTLSv1.2
```

1 2 3

```
⊟ DTLSv1.2 Record Layer: Handshake Protocol: Hello Verify Request
    Content Type: Handshake (22)
    Version: DTLS 1.2 (0xfefd)
    Epoch: 0
    Sequence Number: 0
    Length: 23
  ⊟ Handshake Protocol: Hello Verify Request
      Handshake Type: Hello Verify Request (3)
      Length: 11
      Message Sequence: 0
      Fragment Offset: 0
      Fragment Length: 11
      Version: DTLS 1.2 (0xfefd)
      Cookie Length: 8
      Cookie: 8edc0b55ede31648
```

```
⊞ Internet Protocol Version 6, Src: fe80::4954:97dd:5a71:f3fa, Dst: fe80::4cdb:a9f9:868:7fa2
⊞ User Datagram Protocol, Src Port: 1025 (1025), Dst Port: 19786 (19786)
⊟ Datagram Transport Layer Security
  ⊟ DTLSv1.2 Record Layer: Handshake Protocol: Client Hello
      Content Type: Handshake (22)
      Version: DTLS 1.2 (0xfefd)
      Epoch: 0
      Sequence Number: 1
      Length: 412
    ⊟ Handshake Protocol: Client Hello
        Handshake Type: Client Hello (1)
        Length: 400
        Message Sequence: 1
        Fragment Offset: 0
        Fragment Length: 400
        Version: DTLS 1.2 (0xfefd)
      ⊞ Random
        Session ID Length: 0
        Cookie Length: 8
        Cookie: 8edc0b55ede31648
        Cipher Suites Length: 2
      ⊞ Cipher Suites (1 suite)
        Compression Methods Length: 1
      ⊞ Compression Methods (1 method)
        Extensions Length: 348
      ⊞ Extension: elliptic_curves
      ⊞ Extension: ec_point_formats
      ⊞ Extension: Unknown 256
```

PSKd

Joiner Sequence – Wireshark Decoding

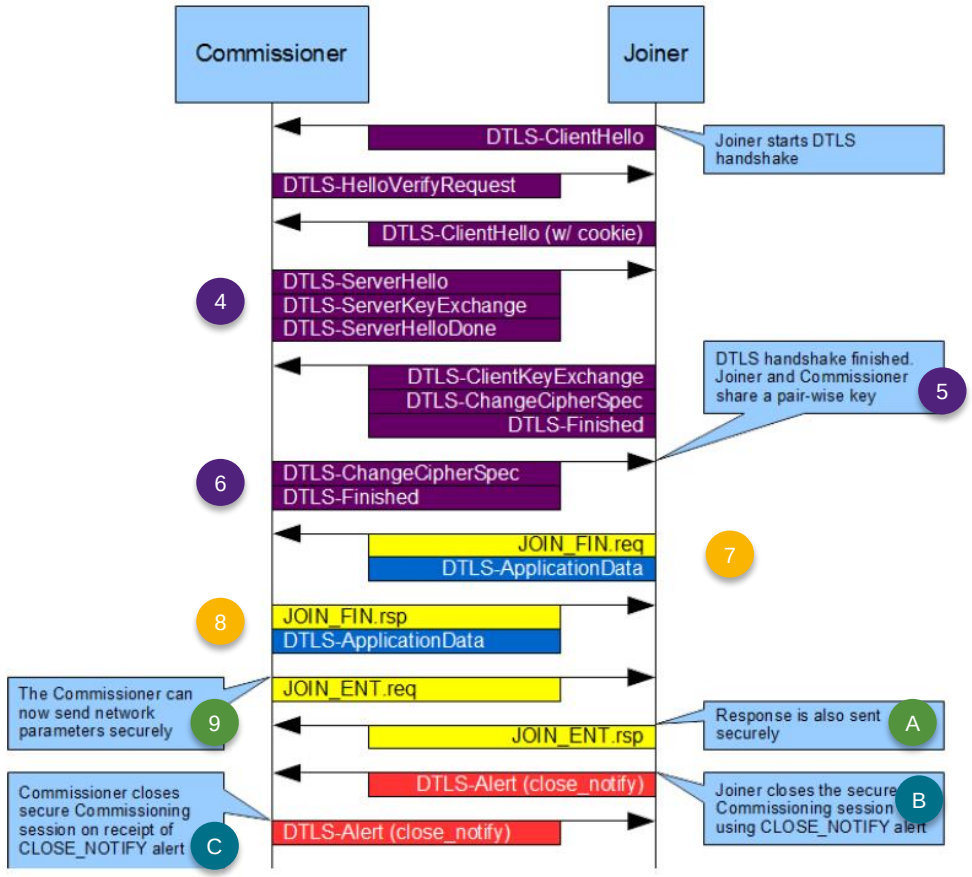


Figure 17. Joiner–Joiner Router/Commissioner Sequence

```
37 Server Hello, Server Key Exchange, Server Hello Done      4   DTLSv1.2
38 Ack                                                           IEEE 802.15.4
39 MLE Secured Advertisement                                     Mesh Link Exchange
40 Data, Dst: 4e:db:a9:f9:08:68:7f:a2, Src: 4b:54:97:dd:5a:71:f3:fa   6LoWPAN
41 Ack                                                           IEEE 802.15.4
42 Data, Dst: 4e:db:a9:f9:08:68:7f:a2, Src: 4b:54:97:dd:5a:71:f3:fa   6LoWPAN
43 Ack                                                           IEEE 802.15.4
44 Client Key Exchange, Change Cipher Spec, Encrypted Handshake Message   DTLSv1.2   5
45 Ack                                                           IEEE 802.15.4
46 Change Cipher Spec, Encrypted Handshake Message               DTLSv1.2   6

50 Application Data                                          7   DTLSv1.2
51 Ack                                                           IEEE 802.15.4
52 Application Data                                          8   DTLSv1.2
53 Ack                                                           IEEE 802.15.4

54 Encrypted Alert                                           B   DTLSv1.2
55 Ack                                                           IEEE 802.15.4
56 Encrypted Alert                                           C   DTLSv1.2
57 Ack                                                           IEEE 802.15.4

57 Ack                                                           IEEE 802.15.4
58 Data, Dst: 4b:54:97:dd:5a:71:f3:fa, Src: 4e:db:a9:f9:08:68:7f:a2   IEEE 802.15.4
59 Ack                                                       9   IEEE 802.15.4
60 Data, Dst: 4b:54:97:dd:5a:71:f3:fa, Src: 4e:db:a9:f9:08:68:7f:a2   IEEE 802.15.4
61 Ack                                                           IEEE 802.15.4
62 Data, Dst: 4e:db:a9:f9:08:68:7f:a2, Src: 4b:54:97:dd:5a:71:f3:fa   IEEE 802.15.4
63 Ack                                                       A   IEEE 802.15.4
```

KINETIS THREAD STACK COMMISSIONING API

Kinetis Thread Stack Commissioning Joiner AP

Just call:

```
thrStatus_t THR_NwkJoin(instanceId_t thrInstId)
```

If `isCommissioned` attribute is set to `FALSE`:

```
bool_t isCommissioned = FALSE;
THR_SetAttr(0, gNwkAttrId_IsDevCommissioned_c, 0, sizeof(bool_t), &isCommissioned);
```

…then `THR_NwkJoin` will attempt Joining via Commissioning:

- Discover if there is an active commissioner
- Discover a Joiner Router
- Initiate DTLS EC-JPAKE to the Border Router

Kinetis Thread Stack MESHCoP Module

API Calls:

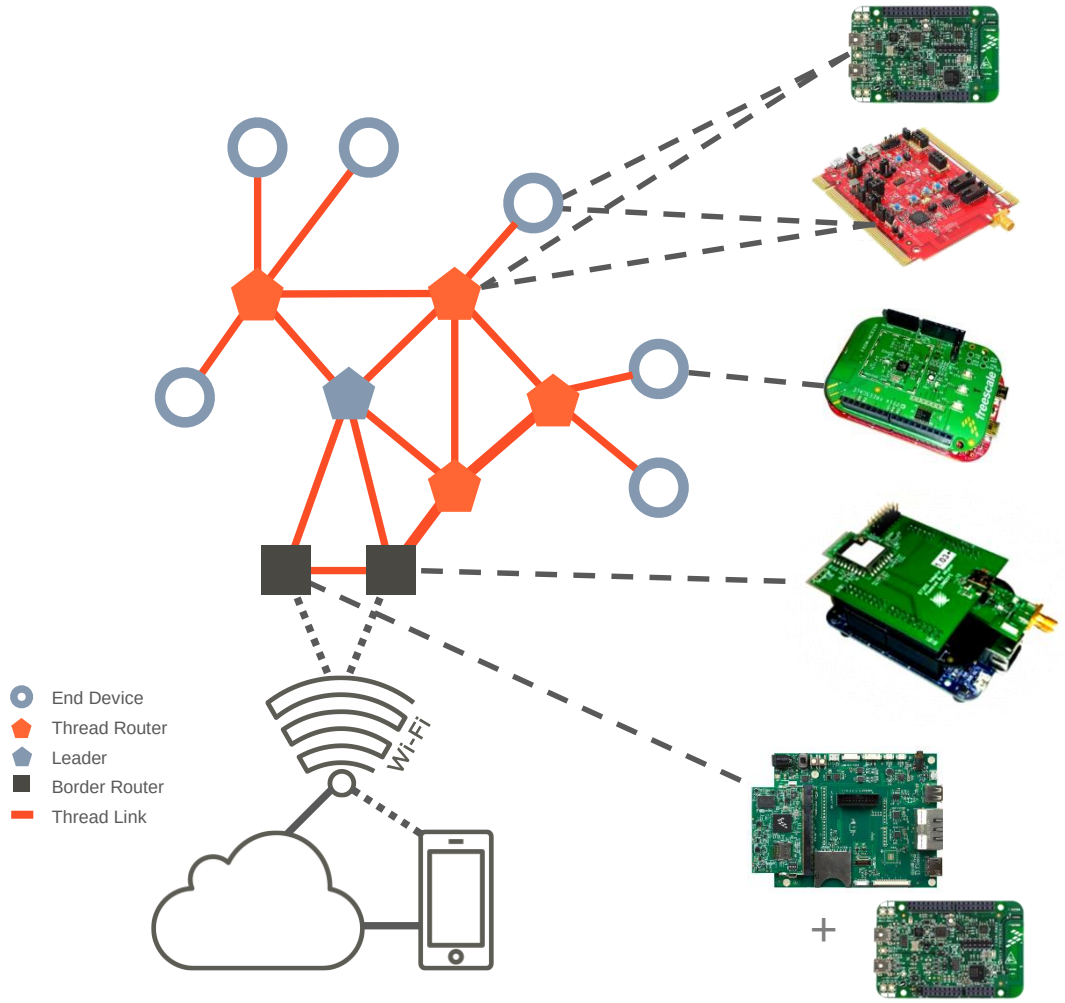
- `void MESHCOP_StartCommissioner(instanceId);`
- `void MESHCOP_StopCommissioner(instanceId);`
- `bool_t MESHCOP_AddExpectedJoiner(instanceId_t thrInstId, uint8_t* pEui, uint32_t euiLen, uint8_t *pPsk, uint32_t pskLen);`
- `bool_t MESHCOP_RemoveExpectedJoiner(instanceId_t thrInstId, uint8_t *pEui, uint8_t euiLen);`
- `void MESHCOP_SyncSteeringData(instanceId_t thrInstId, meshcopEuiMask_t euiMask);`
- `void MESHCOP_RemoveAllExpectedJoiners(instanceId_t thrInstId);`

API Asynchronous Events:

- `gThrEv_MeshCop_LocalCommissionerRejected_c`
- `gThrEv_MeshCop_LocalCommissionerAccepted_c`
- `gThrEv_MeshCop_JoinerAccepted_c`
- `gThrEv_MeshCop_JoinerRejected_c`
- `gThrEv_MeshCop_KeepAliveSent_c`

NXP KINETIS & I.MX THREAD PLATFORM

NXP’s Thread Hardware Offering



NXP Kinetis KW2xD, KW41Z (Upcoming)
Thread Router / REED / End Device
Tower Board, Freedom Board and USB Stick available
Kinetis SDK and FreeRTOS

NXP Kinetis KL46 + MCR20A Transceiver
Thread End Device
Freedom Board
Kinetis SDK and FreeRTOS

NXP Kinetis K64F + MCR20A Transceiver
Border Router with Ethernet & upcoming Wi-Fi support (QCA400x)
Freedom Boards
Kinetis SDK and FreeRTOS

NXP i.MX6 UltraLite EVK + FRDM-KW24D512
Thread Border Router / Cloud gateway
Provides IP data routing and infrastructure integration
i.MX6UL EVK & KW24D FRDM Board (via SPI or UART)
Runs Linux operating system

The most **complete** Thread end to end platform available!

Device Summary for Thread Solution

MCUs

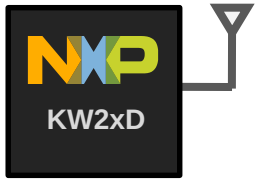
- M0+ @72MHz
- 256K Flash / 32K RAM
- Crypto in Software

- M4 @120MHz
- 1M Flash / 256K RAM
- Integrated Ethernet
- Hardware Crypto

MPUs

- A7 @528MHz
- Linux BSP
- LCD Display upt WXGA (1366x768)
- Consumer to Industrial Temp
- DDR2/DDR3
- Hardware Crypto and Tamper

MCU + Radios



- M4 @50MHz
- 512KF/64KR or 256KF/32KR
- 2.4GHz 15.4-2006 (only)
- -102 dBm sensitivity
- 17mA at 0dBm and Rx 19mA typical
- -35 to + 8dBm output power
- USB Option
- Dual Pan ID / Dual Channel ID in Hdwr
- Fast Switching Antennae Diversity
- Hardware Crypto

Launch Oct'16

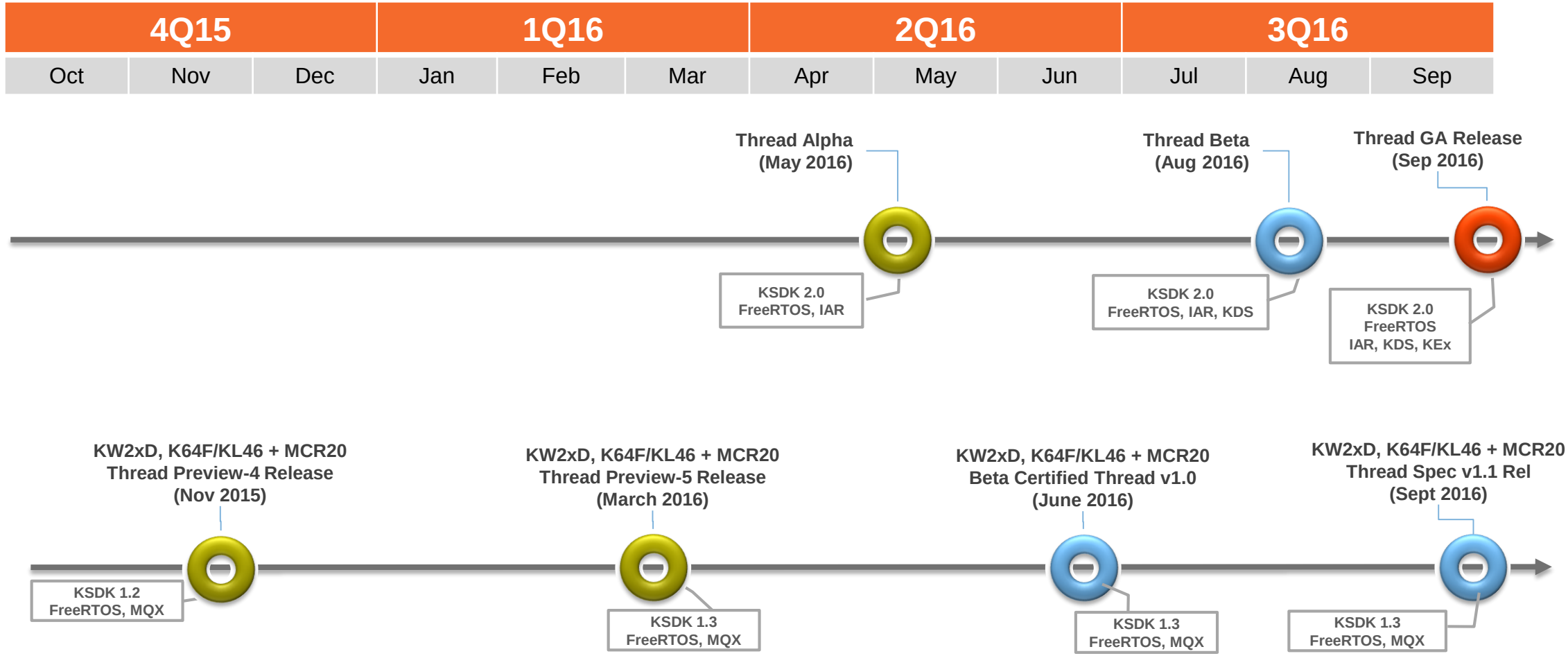
New

- M0+ @48MHz
- 512K Flash/128K RAM or 256K Flash/64K RAM
- 2.4GHz 15.4 (+ BLE v4.2 option – KW41Z)
- -102dBm sensitivity in 15.4, -96dBm in BLE
- 15.4mA Tx/RX at 0dBm (without DC-DC)
- 6.5mA Tx/RX (with DC-DC)
- -25 to +3.5dBm output power
- Integrated balun (~9% board area savings)
- Dual Pan ID / Dual Channel ID in Hdwr
- Fast Switching Antennae Diversity
- Hardware Crypto

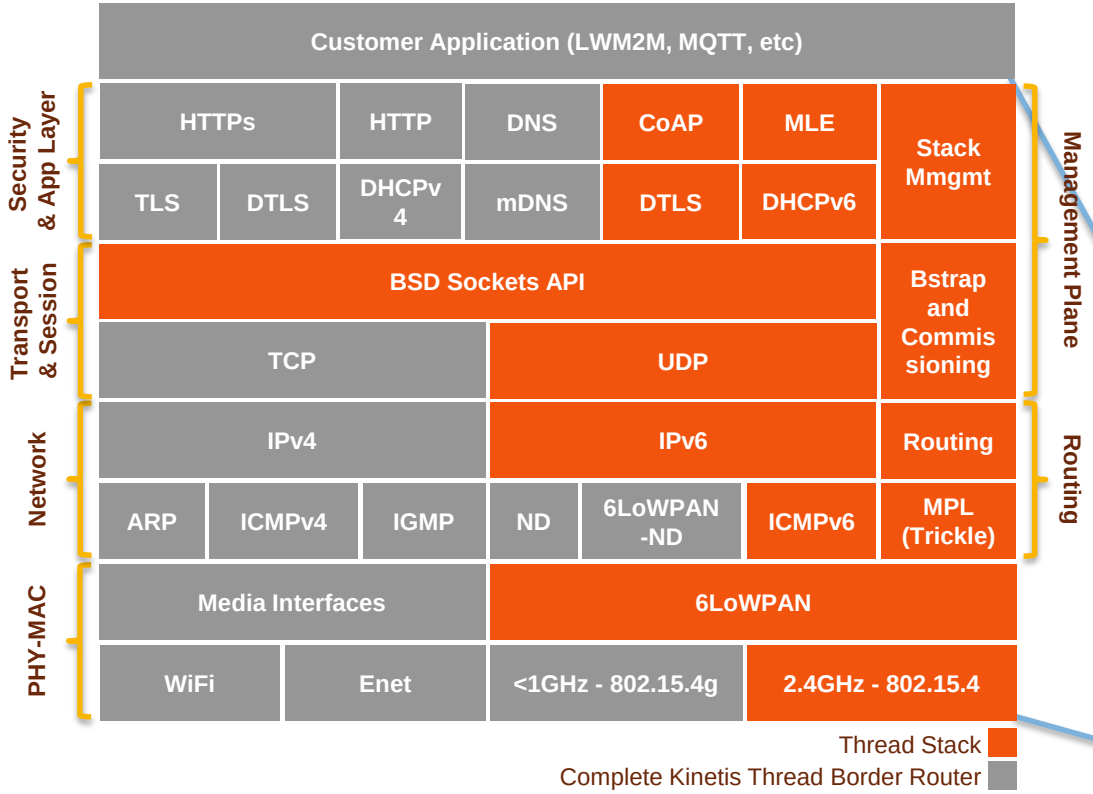
Transceiver Only

- Same radio specs as KW2xD

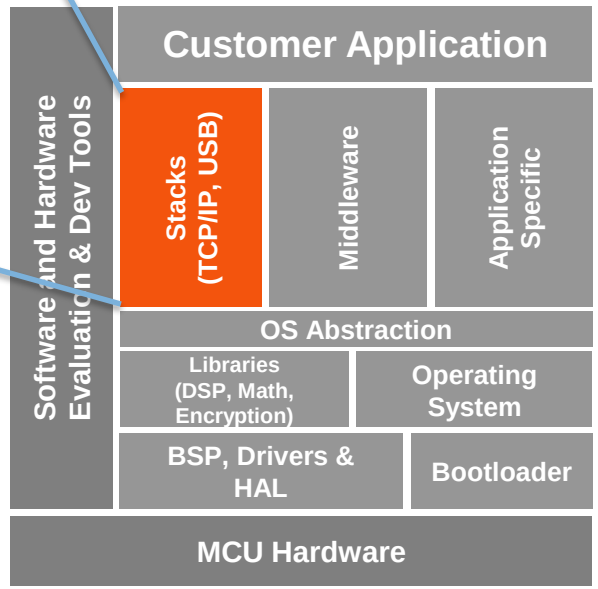
Thread Software Timeline



Kinetis Thread Stack Overview

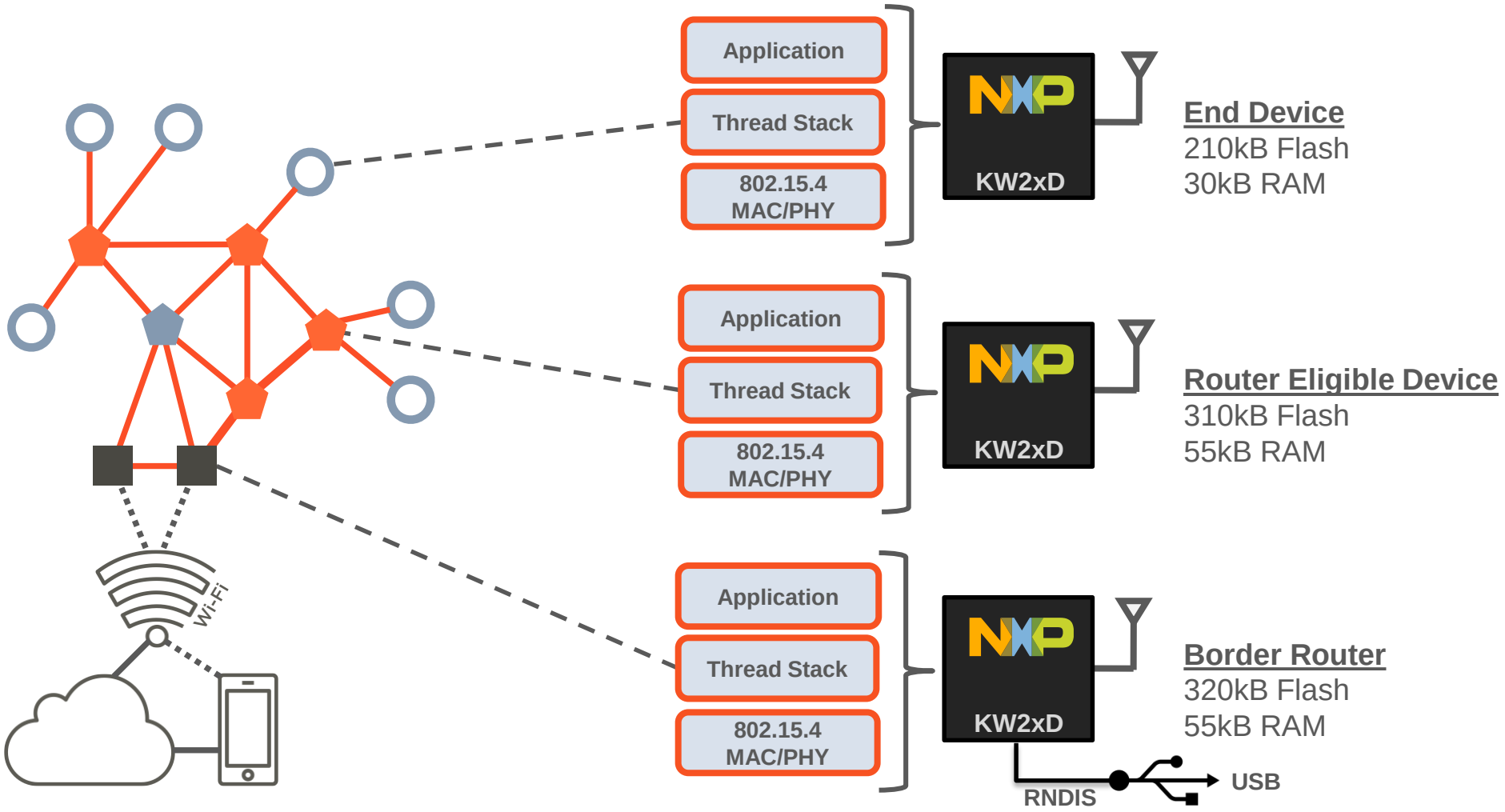


- **Product Features:**
  - Built **on top of Kinetis SDK (1.3)**
  - **Multiple OS support** via Kinetis SDK OSA
  - **Thread stack successfully proven** interoperability with other vendors.



- **Product Features:**
  - **Flexible, configurable and scalable** Dual Stack IPv4 & IPv6 for constrained resources devices
  - **Multiple interfaces support: 802.15.4 & 802.15.4g with 6LoWPAN**, Ethernet and Wi-Fi
  - **Designed for Low Power,** Quick Wake-up Time and Low Memory footprint

KW2xD Thread Device Type Code Estimates



Thread Demo Applications

- **Temp Sensor using CoAP**
  - Sends temp when button is pressed
  - Data encoded using a CoAP frame format at the application layer
  - Sent over UDP at the transport layer

- **Data Sink using CoAP**
  - Set a node to announce itself as an application Data Sink – representing a single concentrator destination node on the network for application messages
  - All CoAP confirmable commands addressed to the Data Sink node will be stored there
  - Set a node to announce releasing (ceasing) the Data Sink role

- **Control RGB LED on FRDM Board using CoAP**
  - Uses board button switches to send an LED ON or LED OFF command to the other devices on the Thread network
  - The LED control command is encoded using a CoAP frame format at the application layer

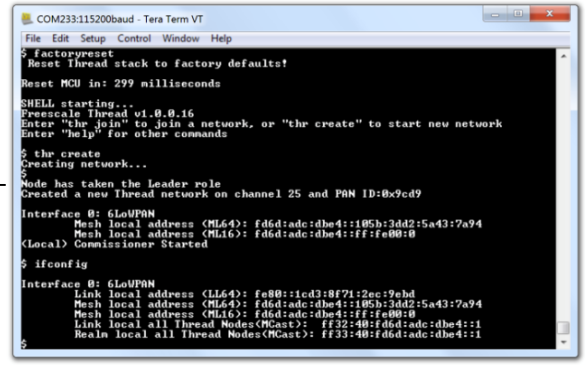
- **Over The Air Firmware Update using CoAP**
  - Client – Server implementation using CoAP unicast messages to update the Thread device firmware to internal or external flash storage

Application Configuration Overview

Example App \ Board	FRDM-KW24D	TWR-KW24D512	TWR-KW21D256	USB-KW24D512	FRDM-KL46Z & FRDM-MCR20A	FRDM-K64F & FRDM-MCR20A
thread_router_eligible_device (template for **mains powered, always-on products driven entirely by Kinetis**: security control panels, standalone sensor hubs, range extenders, smart plugs, some thermostats, wall light switches, some light fixtures, some appliances)	**CoAP**: led, temp, sink **UART**: shell **USB**: N/A **Commission**: auto-start collapsed commissioner **Lib Capability**: leader, router, reed, rxoned, pollinged, lped	**CoAP**: led, temp, sink **UART**: shell **USB**: N/A **Commission:** auto-start collapsed commissioner **Lib Capability**: leader, router, reed, rxoned, pollinged, lped	N/A	**CoAP**: temp, sink **USB**: shell **Commission:** auto-start collapsed commissioner **Lib Capability**: leader, router, reed, rxoned, pollinged	N/A	**CoAP**: led, temp, sink **UART:** shell **ETH**: N/A, **USB**: N/A **Commission:** auto-start collapsed commissioner **Lib Capability**: leader, router, reed, rxoned, pollinged, lped
thread_end_device (template for **mains powered** or **high-capacity battery products driven entirely by Kinetis** which are **NOT intended to be always-on**: light fixtures, appliances, some door locks, some thermostats, some resource constrained devices)	**CoAP**: led, temp, sink **UART**: shell **USB**: N/A **Other**: rxoned default **Lib Capability**: rxoned, pollinged, lped	**CoAP**: led, temp, sink **UART**: shell **USB**: N/A **Other**: rxoned default **Lib Capability**: rxoned, pollinged, lped	**CoAP**: led, temp, sink **UART**: shell **Other**: rxoned default **Lib Capability**: rxoned, pollinged, lped	**CoAP:** temp, sink **USB**: shell **Other**: rxoned default **Lib Capability**: rxoned, pollinged	**CoAP:** led, temp, sink **UART**: shell **USB**: N/A **Other**: rxoned default **Lib Capability**: rxoned, pollinged, lped	**CoAP:** led, temp, sink **UART**: shell **ETH**: N/A, **USB**: N/A **Other**: rxoned default **Lib Capability**: rxoned, pollinged, lped
thread_low_power_end_device (template for **low-capacity battery Kinetis products**: sensors, remote controls, fobs, door locks)	**CoAP:** temp, sink **UART**: N/A **USB**: N/A **LP**: LP mode 10 **Lib Capability**: lped	**CoAP:** temp, sink **UART**: N/A **USB**: N/A **LP**: LP mode 10 **Lib Capability**: lped	**CoAP:** temp, sink **UART**: N/A **LP**: LP mode 10 **Lib Capability**: lped	**CoAP:** temp, sink **USB**: N/A **LP**: LP mode 10 **Lib Capability**: lped	**CoAP:** temp, sink **UART**: N/A **USB**: N/A **LP**: LP mode 10 **Lib Capability**: lped	**CoAP:** temp, sink **UART**: N/A **USB**: N/A **LP**: LP mode 10 **Lib Capability**: lped
thread_border_router (template for all product categories above which use a standalone **Kinetis to forward IP packets from/to the Thread subnet** and an alternate IP capable interface working via Ethernet, Wi-Fi, or USB – in order to establish local network or Internet end-to-end IP connectivity)	**CoAP:** led, temp, sink **UART**: shell **USB**: ND_ROUTER over RNDIS, no THCI **Commission:** auto-start collapsed commissioner **Lib Capability**: leader, router, reed, rxoned, ipv4, nd	**CoAP:** led, temp, sink **UART**: shell **USB**: ND_ROUTER over RNDIS, no THCI **Commission:** auto-start collapsed commissioner **Lib Capability**: leader, router, reed, rxoned, ipv4, nd	N/A	**CoAP:** temp, sink **USB**: ND_ROUTER and THCI over RNDIS **Commission:** auto-start collapsed commissioner **Lib Capability**: leader, router, reed, rxoned, ipv4, nd	N/A	**CoAP:** led, temp, sink **UART**: shell **ETH**: ND_ROUTER **USB**: N//A **Commission:** auto-start collapsed commissioner **Lib Capability**: leader, router, reed, rxoned, ipv4, nd
thread_host_controlled_device (template for **products where a Kinetis running the Thread stack is hosted by an application processor** over UART or SPI; use of Thread Host SDK tools is recommended for HLOS UNIX host systems; serves as sub-component for advanced asymmetric multiple chip **border routers**)	**CoAP:** led, temp, sink **UART**: THCI over FSCI **Serial TUN:** enabled **USB**: N/A **Lib Capability**: leader, router, reed, rxoned, ipv4, nd	**CoAP:** led, temp, sink **UART**: THCI over FSCI **Serial TUN:** disabled **USB**: N/A **Lib Capability**: leader, router, reed, rxoned, ipv4, nd	N/A	**CoAP:** temp, sink **USB**:THCI over FSCI, **Serial TUN:** enabled **Lib Capability**: leader, router, reed, rxoned, ipv4, nd	N/A	**CoAP:** led, temp, sink **UART:** THCI over FSCI **Serial TUN:** disabled **ETH**: N/A; **USB**: N/A **Lib Capability**: leader, router, reed, rxoned, ipv4, nd

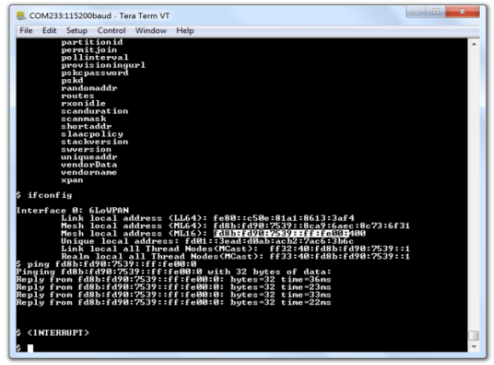
Kinetis Embedded Demos for Self Contained Networks

Router Eligible Devices



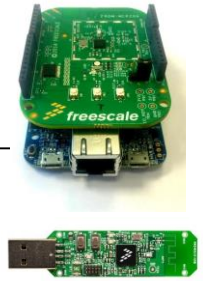
- CLI interface for REED and End Nodes
- Human readable commands implemented for network creation, commissioning, configuration, testing
- Push button demo to send Node temperature multicast to all nodes
- Simple push button lighting demo.

REED or End Node or Low-Power End Node



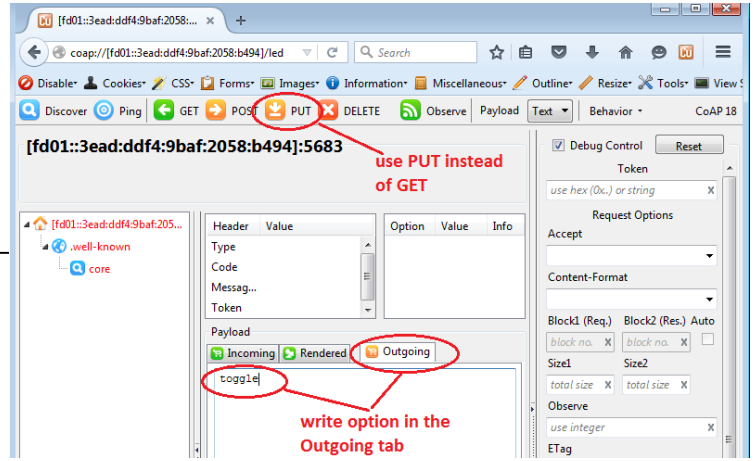
- CLI interface for REED and End Nodes
- Human readable commands implemented for setup nodes for commissioning, join network, network configuration, testing
- Push button demo to send Node temperature multicast to all nodes
- Simple push button lighting demo.

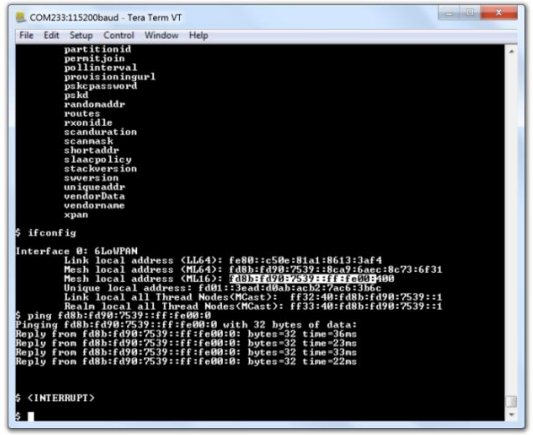
Kinetis Ethernet Border Router – Out of Box Demos



K64F + MCR20A
Ethernet Border Router
or
KW2x USB RNDIS
Border Router

KW2x REED or
End Node





- Communicate with Thread devices from Windows via Ethernet or USB RNDIS interfaces
- CoAP (get/post) examples for remote temperature and LED control/toggle using Copper Firefox add-on

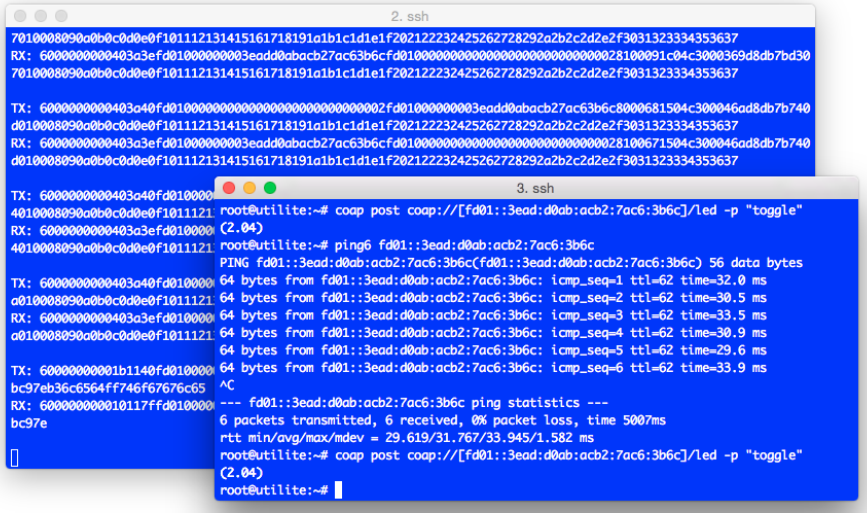
- CLI interface for REED and End Nodes
- Human readable commands implemented for network creation, commissioning, configuration, testing
- Push button demo to send Node temperature multicast to all nodes
- Simple push button lighting demo.

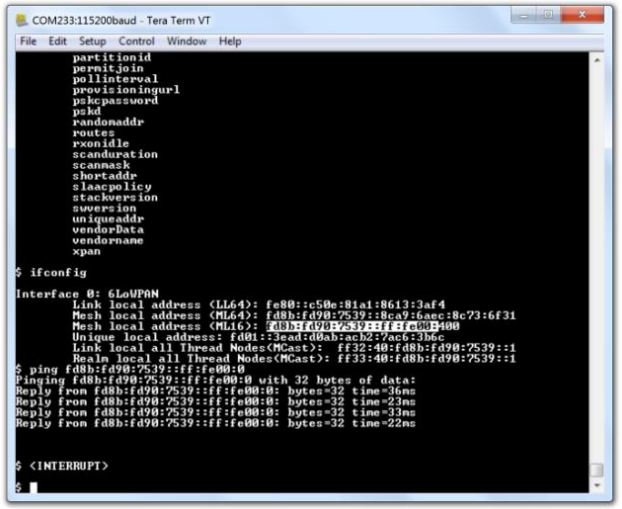
i.MX6 Linux Border Router + THCI – Out of Box Demos

i.MX6 Yocto Linux

FRDM-24x THCI

KW2x REED or
End Node

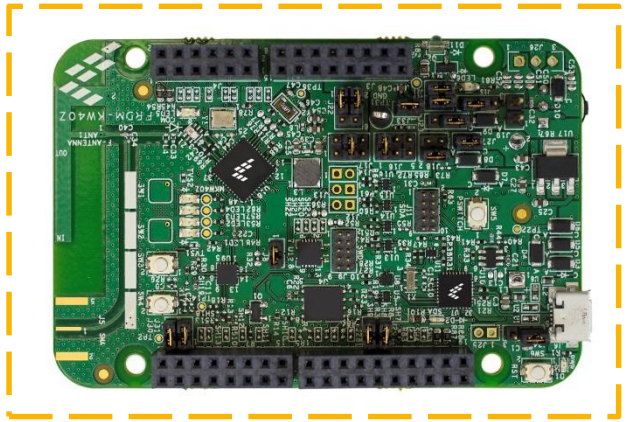




- Linux Host control interfaces
- C and Python Based Virtual TUN Tap drivers for IPv6 packet routing
- Commissioning example code
- CoAP (get/post) examples for remote temperature and LED control/toggle

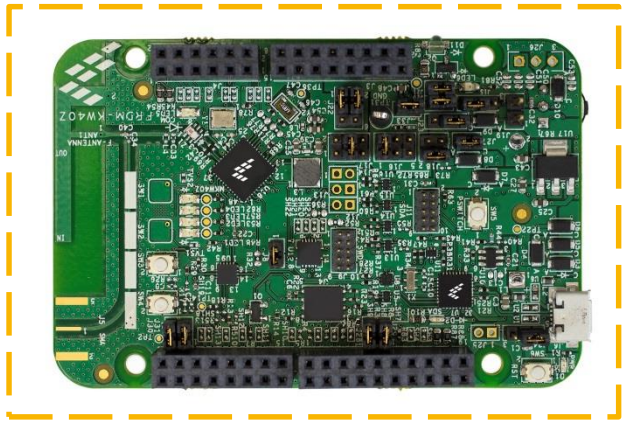
- CLI interface for REED and End Nodes
- Human readable commands implemented for network creation, commissioning, configuration, testing
- Push button demo to send Node temperature mutlicast to all nodes
- Simple push button lighting demo.

Kinetis Thread - Over The Air Firmware Update

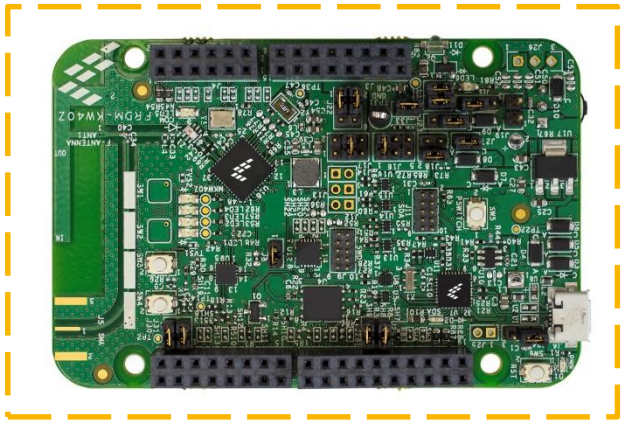


OTAP Client

CoAP transfers



OTAP Client



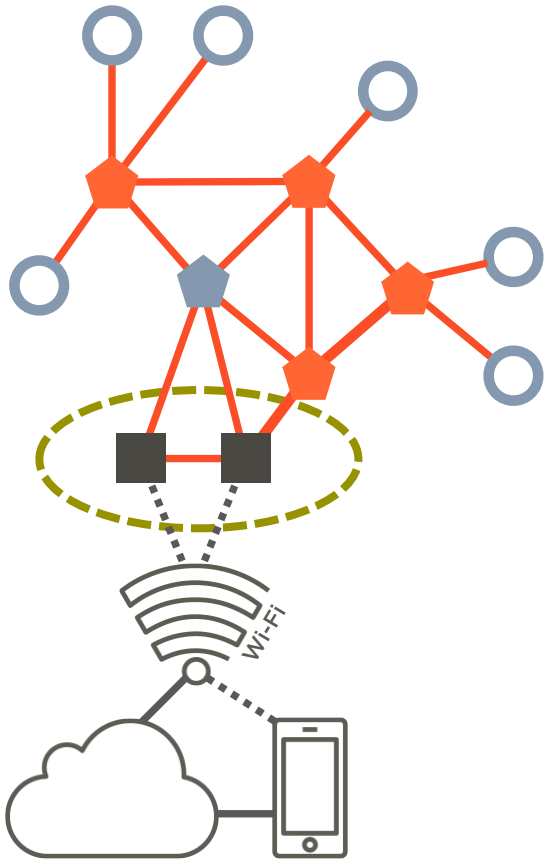
OTAP Client

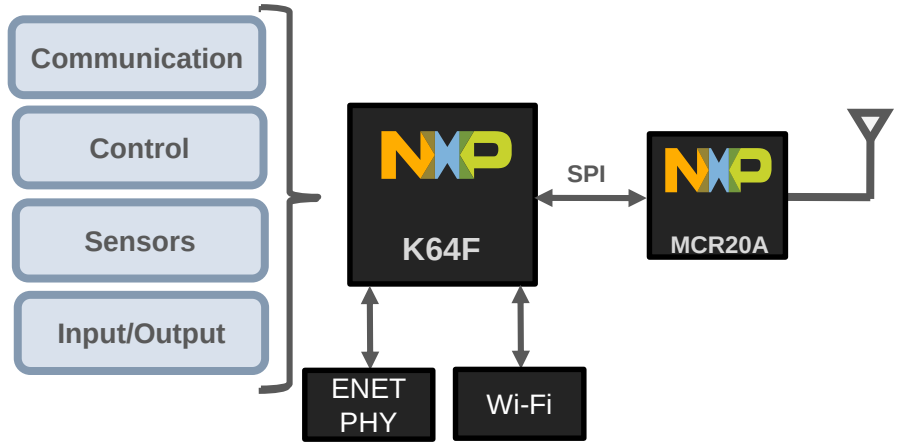
NXP THREAD BORDER ROUTER OPTIONS

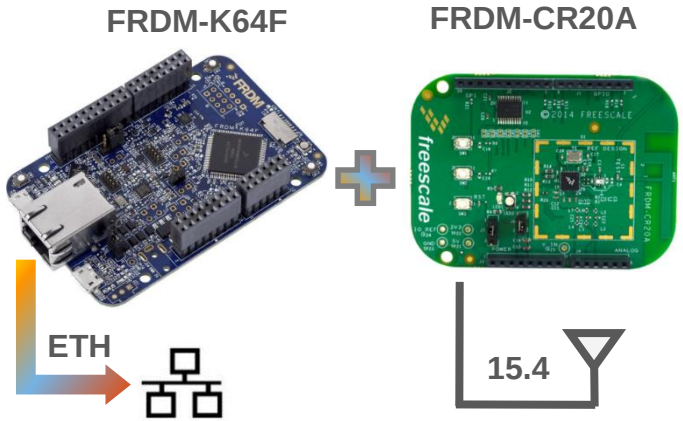
Thread MCU (RTOS) Border Router

K64 is a standalone MCU with up to 1MB Flash, up to 256kB RAM and embedded Ethernet
Memory configuration can support Thread stack, Ethernet stack and Application
MCR20A is a 2.4GHz 802.15.4 transceiver

Wi-Fi (Qualcomm Atheros QCA400x) support in late Q2.







Kinetis K64F/63 120MHz MCUs

Core/System

- Cortex-M4 up to 120MHz with FPU

Memory

- up to 1MB Flash
- up to 256kB SRAM
- up to 4kB EEPROM (FlexMemory)

Communications

- USB OTG FS/LS w/ PHY and USB Vreg
- Ethernet w/ IEEE1588
- CAN
- Multiple serial ports

Analog

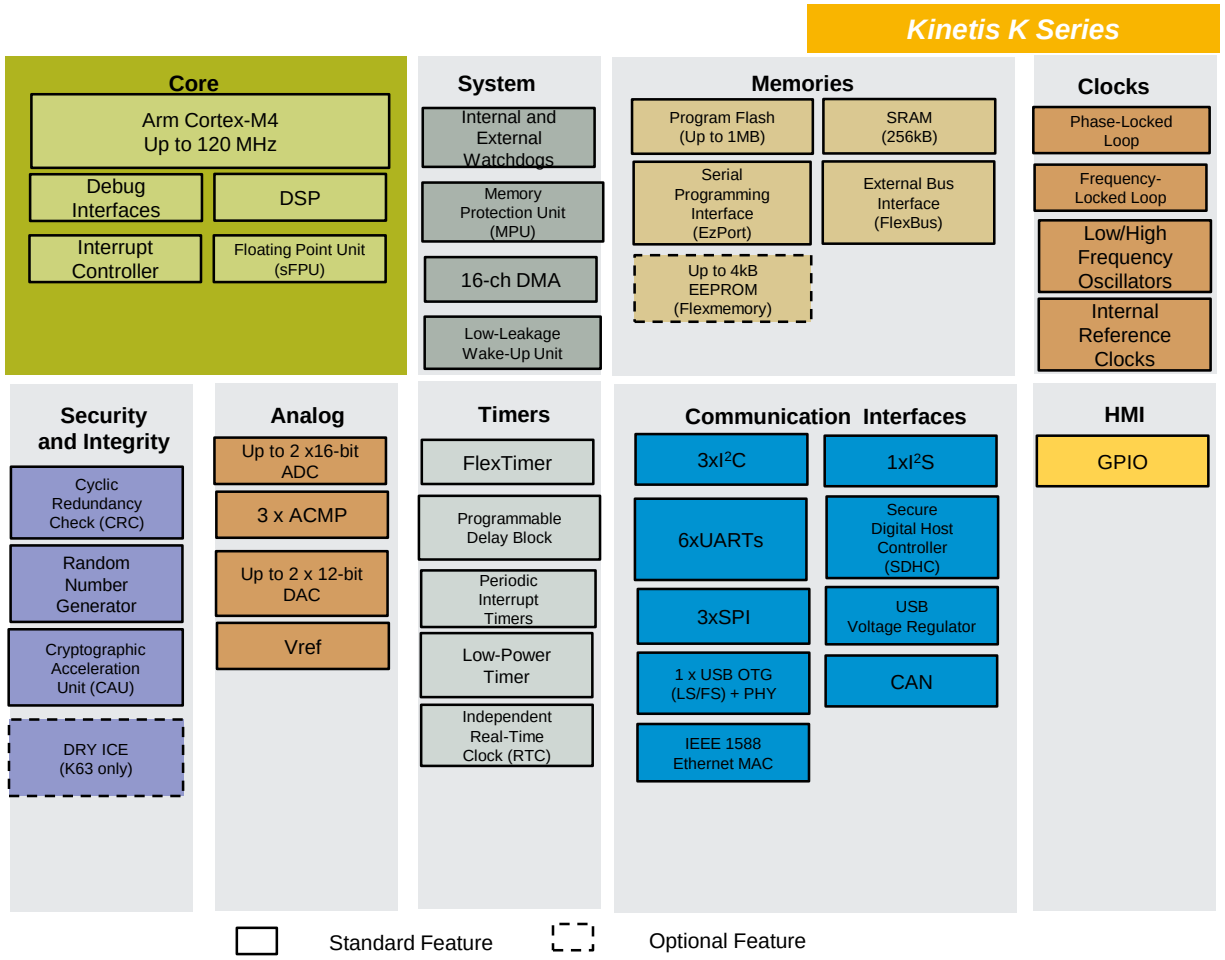
- 2x 16-bit ADC
- 2 x 12-bit DAC; 3 x ACMP

Timers

- 2x8ch FTM (PWM)
- 2x2ch FTM (PWM/Quad Dec.)
- Low Power Timer
- RTC with independent Vbat supply

Others

- 1.71V-3.6V; -40 to 105oC
- Tamper and Crypto acceleration
- DRY ICE (K63F only)



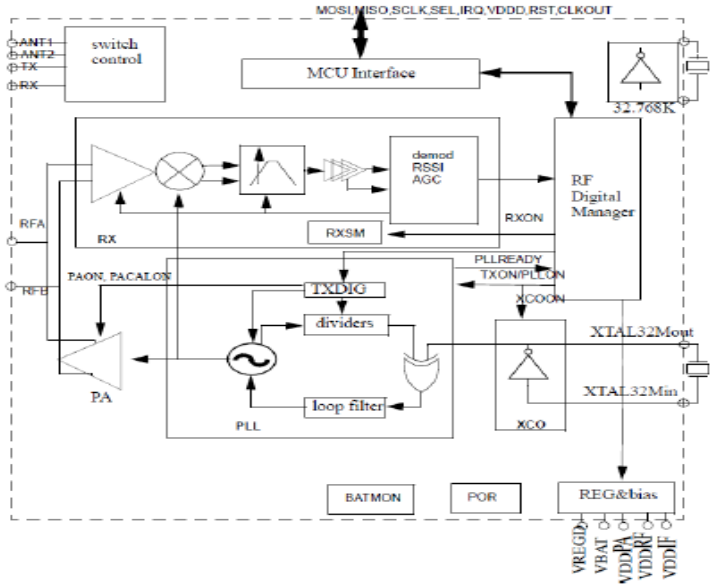
MCR20A High-Performance 802.15.4 Transceiver

2.4 GHz Radio Transceiver Features

- High performance 2.4 GHz IEEE 802.15.4 RF transceiver
- Support for MBAN frequencies (2.36-2.4 GHz)
- Packet processor for hardware acceleration
- Supports single ended and diversity antenna options
- Dual-PAN support
- -30 to + 8 dBm power output
- Support for external PA/LNA (FEM)
- -102 dBm sensitivity
- Tx 17mA @ 0dBm
- Rx 15mA LPPS mode, 19mA full Rx
- AES Hardware encryption/decryption
- True Random Number Generator
- SPI Interface (memory mapped)
- 6 GPIO

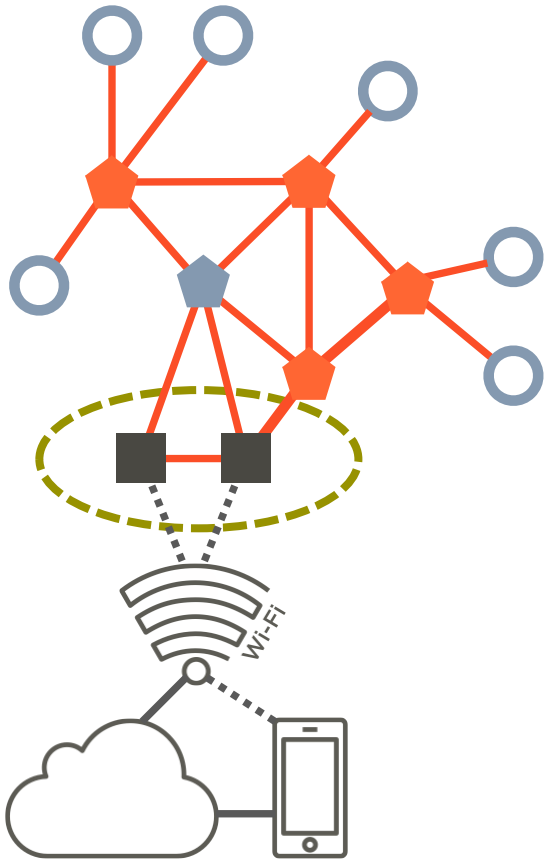
System Features

- -40°C to 105°C
- Operating range: 1.8 V to 3.6 V, -40C to +105C
- 5x5 32-pin QFN

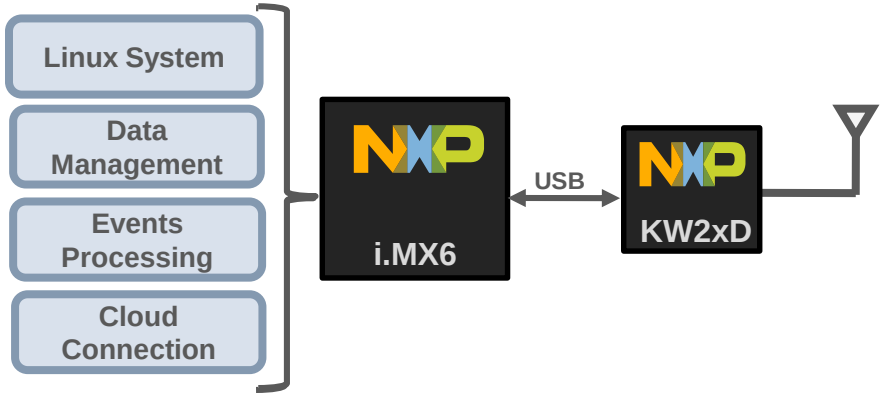


Ordering Part Number: **MCR20AVHM**

Thread MPU (OS) Border Router



KW2xD device runs the Thread Border Router functionality while the i.MX6 Linux system handles Data Management and Analytics, Events Processing and Cloud Connection



i.MX6UL

FRDM-KW24D512

i.MX 6 Series: Supreme Scalability and Flexibility

Scalable series of **NINE** ARM-based SoC Families

i.MX 6UltraLite Family	i.MX 6SoloLite Family	i.MX 6SoloX Family	i.MX 6Solo Family	i.MX 6DualLite Family	i.MX 6Dual Family	i.MX 6DualPlus Family	i.MX 6Quad Family	i.MX 6QuadPlus Family

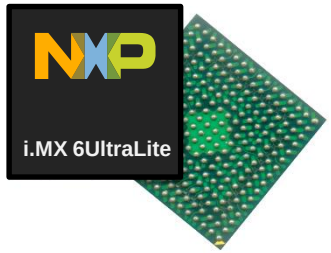
Pin-to-pin Compatible

Software Compatible

i.MX 6UltraLite Advantages

Lowest cost and smallest i.MX 6 member

- ARM Cortex- A7 @ 528 MHz



- The 14x14 289 MAPBGA with 0.8mm pitch for simple and low cost PCB design.

- The 9x9 289 MAPBGA with 0.5mm pitch for space constrained applications.

Most Power efficient Applications Processor

- Integrated power management module that reduces the complexity of external power supply and simplifies power sequencing.

"It provides up to 20% more single thread performance than the Cortex-A5 and provides similar performance to mainstream Cortex-A9 based smartphones in 2012 while consuming less power."
www.arm.com/products/processors/cortex-a/cortex-a7.php

Connectivity optimized for Industrial and IoT applications

- 2x high-speed USB on-the-go with PHY
- Multiple expansion card ports (high-speed)
- 2x 12-bit ADC modules (up to 10 input channels)
- 2x smart card interfaces compatible with EMV Standard v4.3 and a variety of other popular interfaces
- 2x CAN ports

Advanced Security

- Hardware-enabled security features that enable secure e-commerce, digital rights management (DRM), information encryption, On-The-Fly DRAM encryption, secure boot and secure software downloads

Border Router – Ethernet/THCI/RNDIS on Linux and/or PC

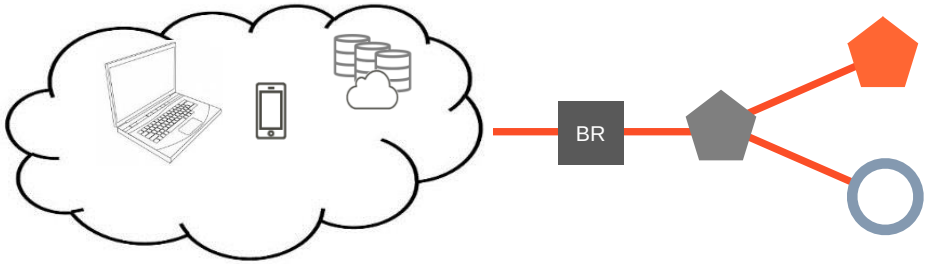
The border router is a device that provides connectivity of nodes in the Thread Network to other devices in external networks (LAN, Internet, VPN)

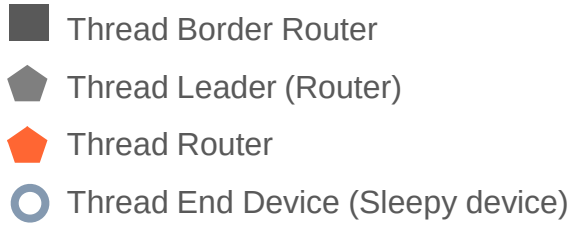
Current development package enables two starting points for Border Router development:

- **Thread Border Router app** running on the FRDM-K64F + FRDM-CR20A
  - Provides external IP connectivity through FRDM-K64F's Ethernet port
  - Static route must be added to the connected PC in order to reach the Thread nwk
  - DHCPv6 from Thread router to the PC is not supported

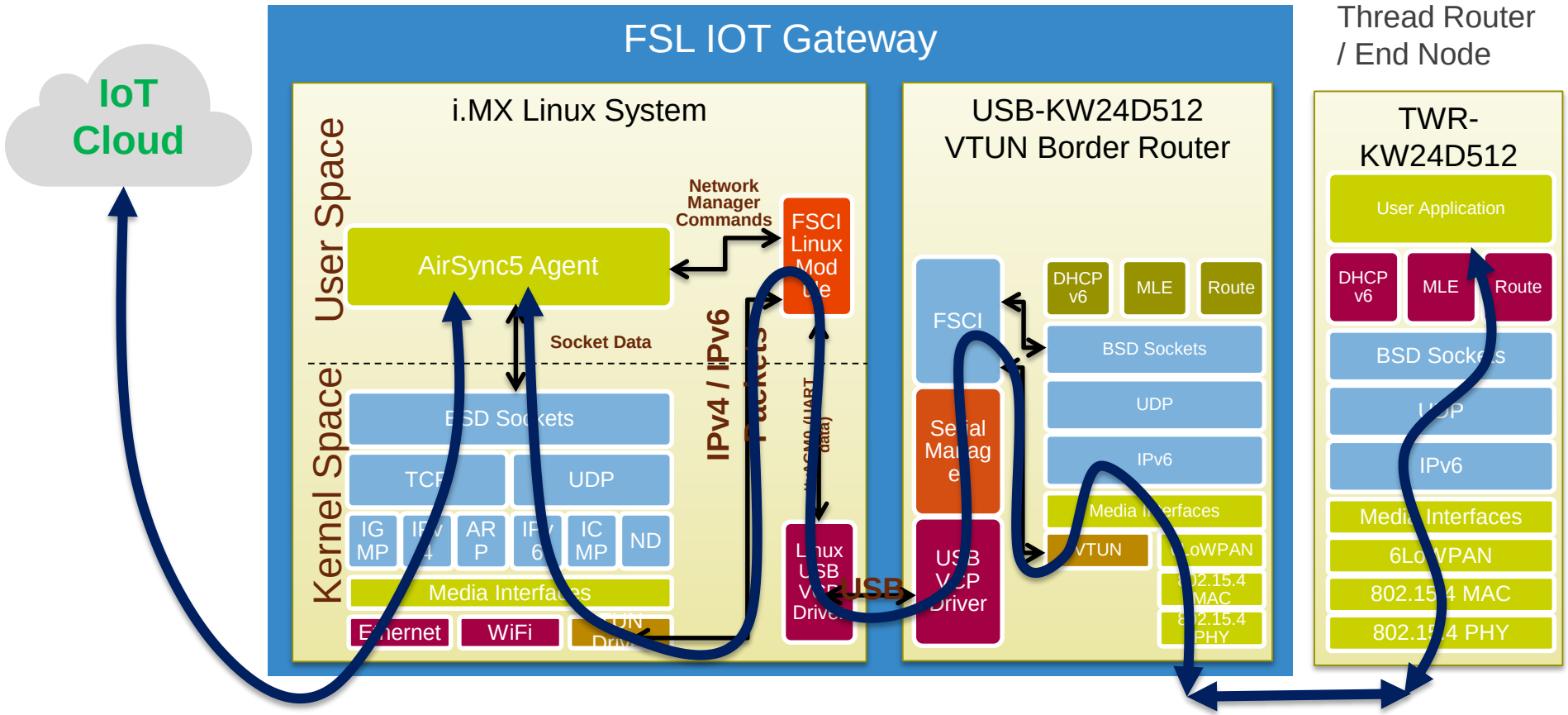
- **Thread Host Controlled Device app** (multiple boards supported)
  - Provides a Virtual Tunnel (VTun) interface that creates an IP layer link over the serial connection to a system running a high level operating system (Linux or Windows) through FSCI commands. See Kinetis-W Host API for Thread Stack.pdf

- **Thread RNDIS app** (multiple boards supported)
  - Using the on-chip USB features of KW2xD to emulate a RNDIS "Ethernet over USB" emulated Network Interface Card for a PC running a host OS such as Windows. IPv6 capable applications running on the host will traverse the Thread network boundary to address Thread network devices end-to end at the IP layer.

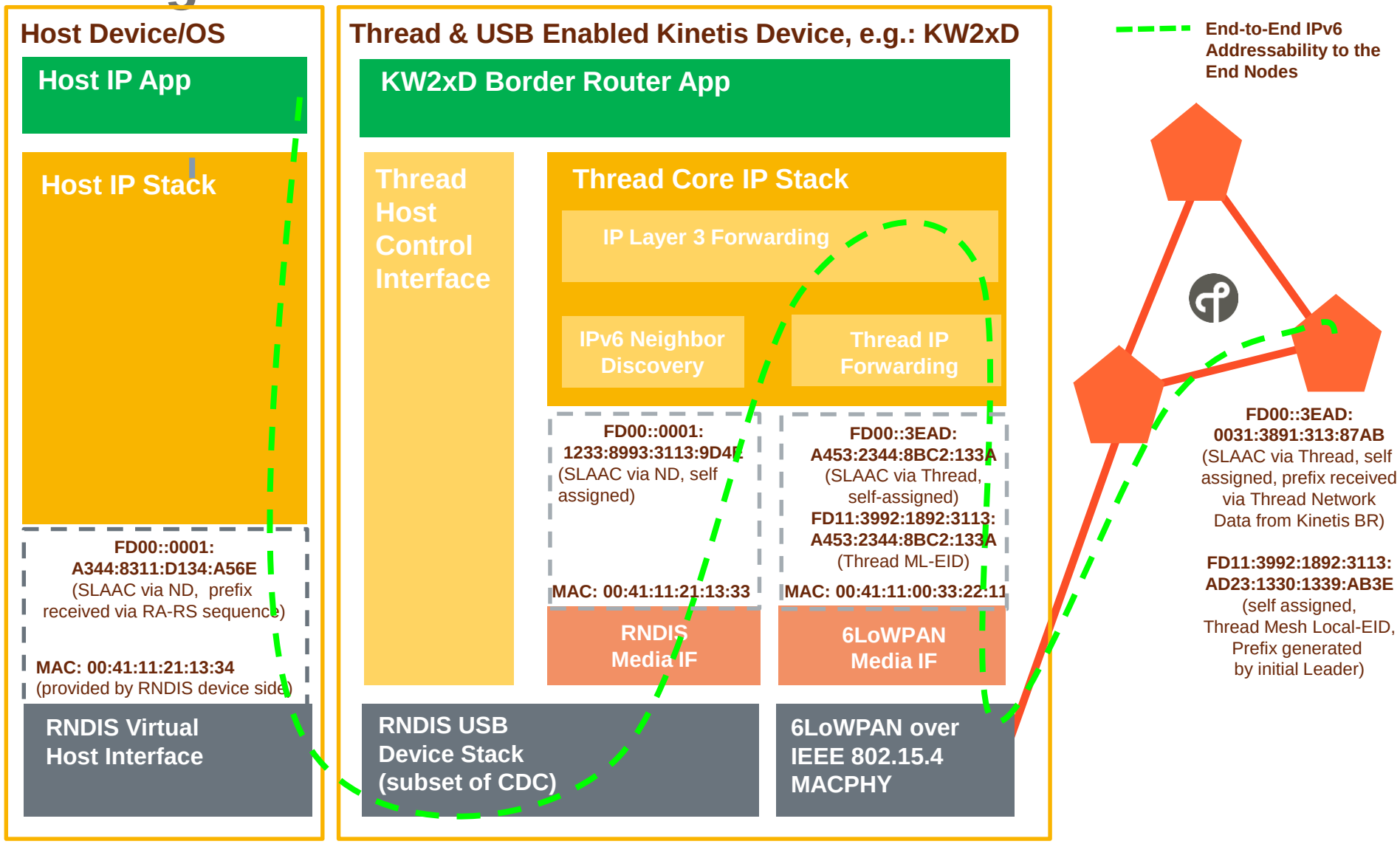




Example Usage of Thread with FSCI and TUN Driver in a Linux i.MX System Paired with Kinetis KW2x



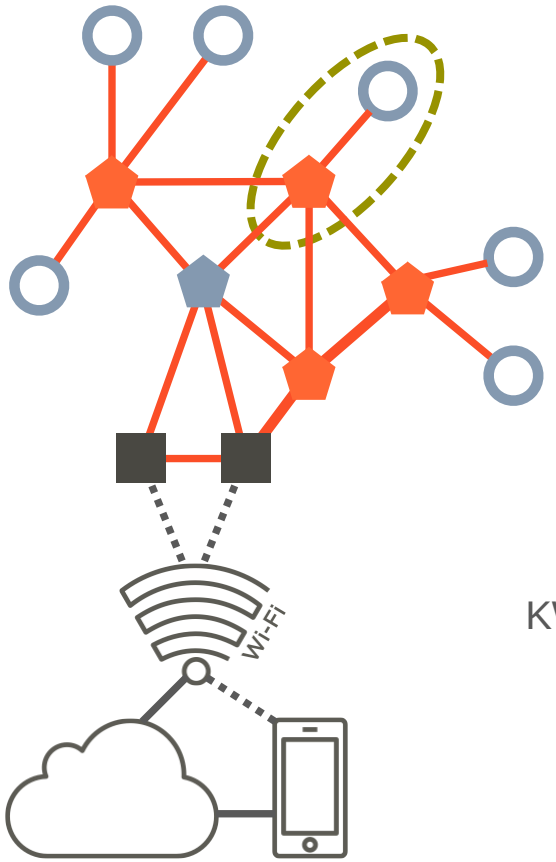
Example usage of Thread with RNDIS

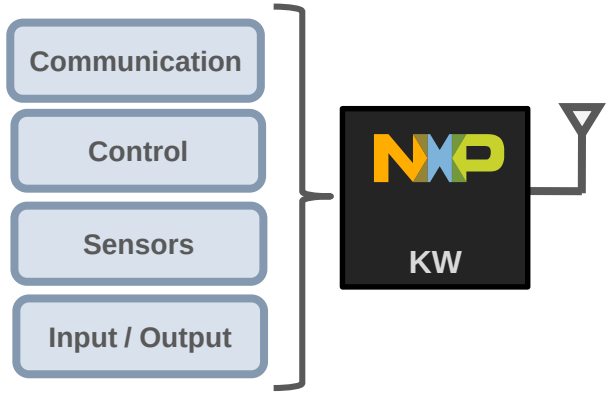


FD00::0001:: – Default Prefix set on the RNDIS link, generated and advertised by KW2xD as ND router
FD00::3EAD:: – Default "global" On-Mesh Prefix assigned by to the Thread subnet
FD11:3992:1892:3113:: – Randomized Mesh Local-only Prefix used on the Thread subnet

NXP THREAD LOW POWER END DEVICES

Thread Router and Low Power End Device



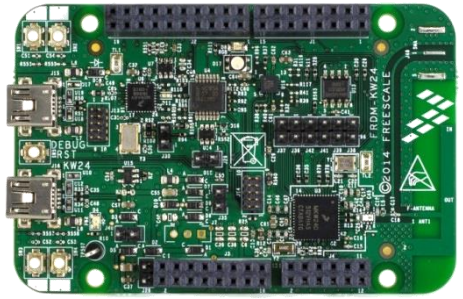


KW devices with 512kB Flash and 64k RAM can run Border Router or Router Eligible Device configurations with an Application

KW devices with 32kB RAM can run Thread End Device configurations with an Application

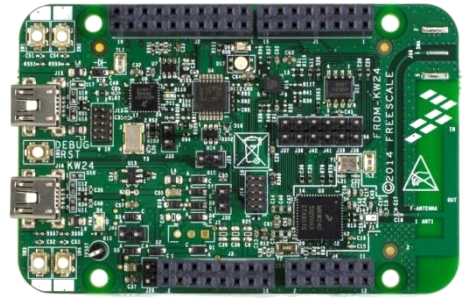
NXP Thread Router and End Node Development Hardware

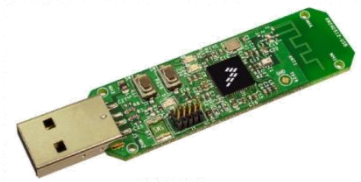
FRDM-KW24D512



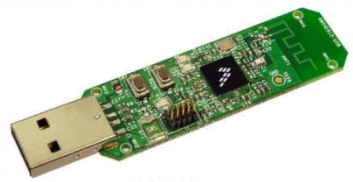
Thread Border Router, Router or End Device

FRDM-KW41D512





Thread Border Router, Router, End Device or Packet Sniffer



USB-KW24D512

AVAILABLE NOW

USB-KW41D512

AVAILABLE 4Q16
THREAD + BLE

Kinetis KW2xD Wireless MCU

Core/System

- Cortex-M4 running up to 50 MHz
- Up to 512kB Flash & 64kB SRAM
- Optional (MKW21D256): 64 kB FlexNVM & 4 kB FlexRAM

2.4 GHz Radio Transceiver

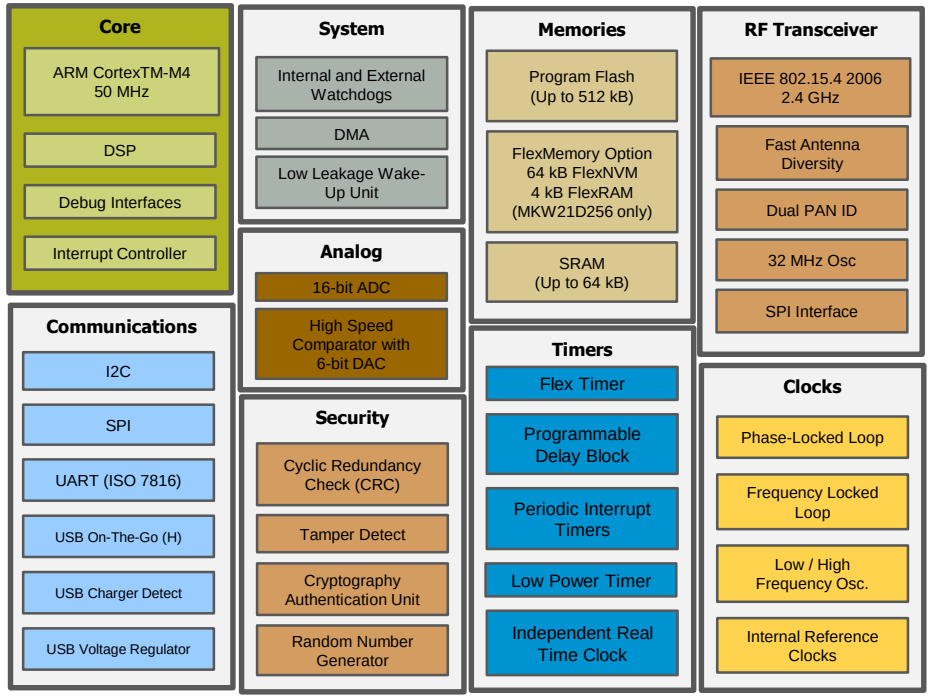
- IEEE-802.15.4 compliant
- -102 dBm Rx sensitivity
- Up to +8dBm Tx output power
- Peak typical current: 17mA Tx and 19mA Rx
- Dual Personal Area Network (PAN) support in hardware
- Run two RF networks simultaneously
- Antenna diversity with automatic antenna selection

Security

- Active and passive tamper detection with RTC timestamp
- Crypto engine: DES, 3DES, AES 128-256, SHA-1, SHA-256, MD5, RNG

System

- UART, SPI, I2C
- Optional USB 2.0 FS/LS H/D/OTG
- 16-bit ADC, 6-bit DAC
- Operating range: 1.8 V to 3.6 V, -40C to +105C



Device	Flash	RAM	Feature	Package
MKW21D256VHA5	256 kB	32 kB	No USB	8x8 63-pin LGA
MKW21D512VHA5	512 kB	64 kB	No USB	8x8 63-pin LGA
MKW22D512VHA5	512 kB	64 kB	USB	8x8 63-pin LGA

Kinetis KW41Z/21Z

AVAILABLE 4Q16

Core/Memory/System
- Cortex-M0+ running up to 48 MHz
- Up to 512 kB Flash, Up to 128 kB SRAM
- Four independently programmable DMA controller channels

2.4 GHz Radio Transceiver
- Support for BLE v4.2, 802.15.4
- -96 dBm in BLE mode, -100 dBm in 802.15.4 mode
- -30 to +4 dBm programmable output power
- Increased coexistence performance
- 6.5 mA Rx & 6.5 Tx (0dBm) current target (DC-DC enabled)
- <2uA low power current
- Integrated balun (~9% board area savings)

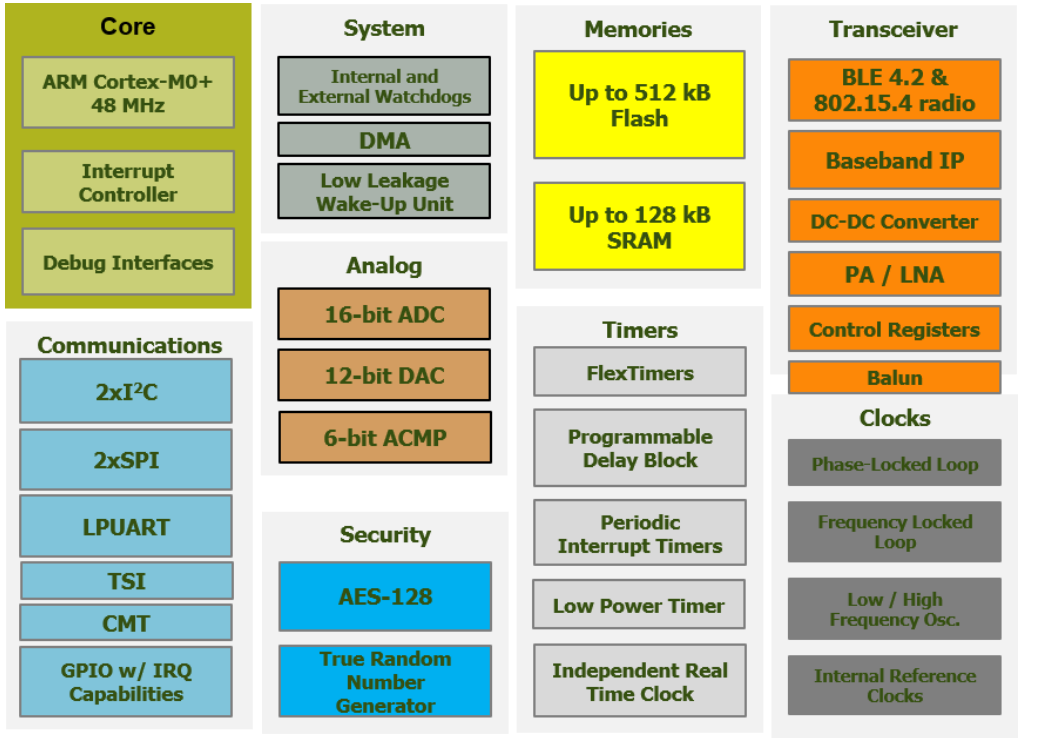
Communications/HMI/Timers
- 2xSPI, LP-UART, 2xI2C, CMT, GPIO with IRQ capability (KBI)
- Hardware Touch Sensing Inputs (TSI)
- 3xFlexTimer (TPM) with PWM & quadrature decode support
- Low Power (LPTMR), Programmable Interrupt (PIT) and RTC timers

Analog
- 16-bit ADC with integrated temperature sensor and battery monitor
- 12-bit DAC and 6-bit High-speed Comparator

Security
- AES Accelerator and True Random Number Generator

Integrated DC/DC Converter
- Normal: 1.71V to 3.6V
- Buck : 2.1V to 4.2V for coin cell operation
- Boost : 0.9V to 1.795V for single alkaline battery operation

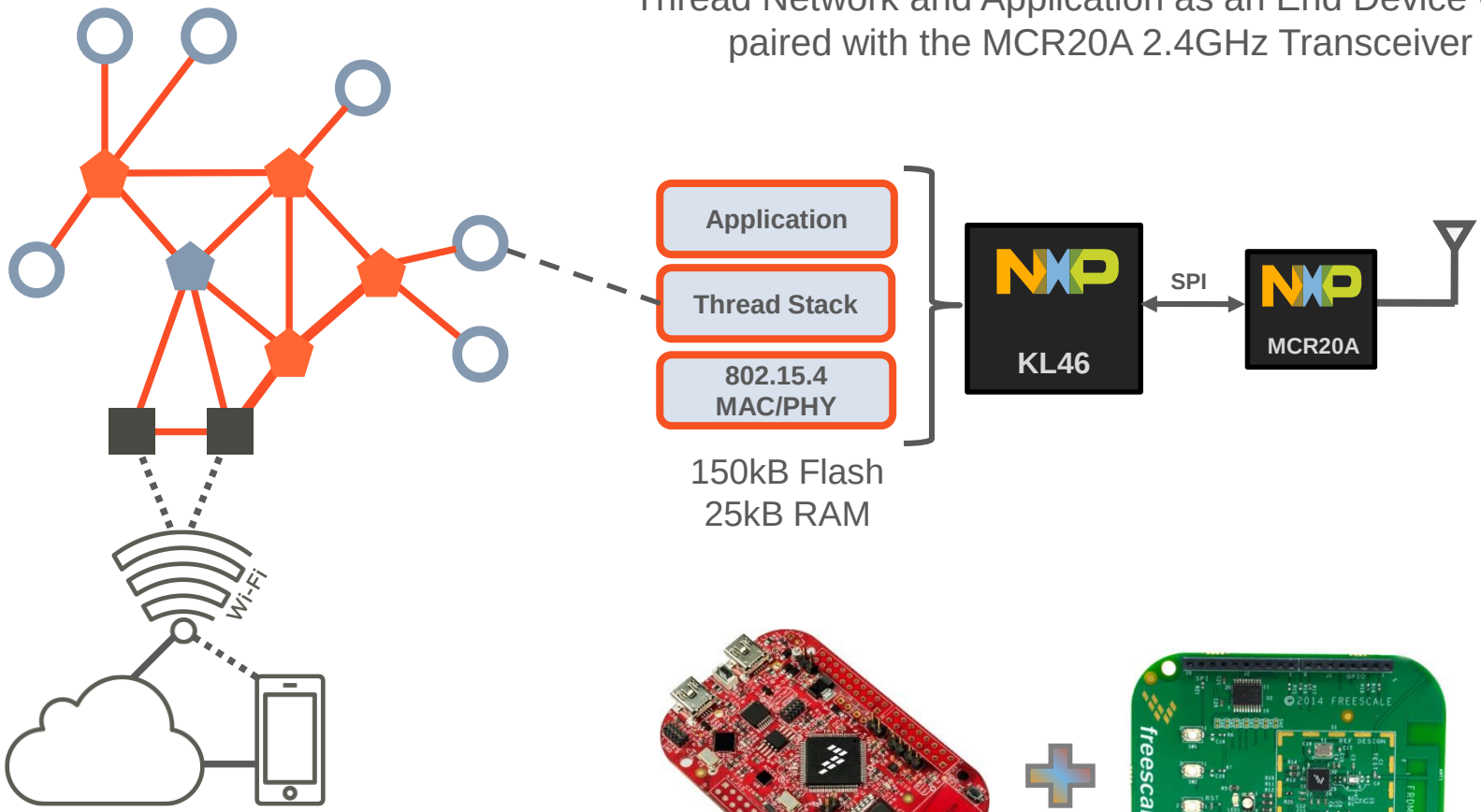
Unique Identifiers
- 80-bit device ID programmed at factory
- 40-bit unique number can be used for Bluetooth Low Energy or IEEE 802.15.4 MAC Address



Device	Memory	Protocol	Package
MKW21Z512VHT4/R **MKW21Z256VHT4/R**	512K Flash, 128K RAM 256K Flash, 64K SRAM	802.15.4	7x7 48-pin Laminate QFN
MKW41Z512VHT4/R **MKW41Z256VHT4/R**	512K Flash, 128K RAM 256K Flash, 64K SRAM	BLE & 802.15.4	7x7 48-pin Laminate QFN WLCSP (PYW)
Features	**Description**		
Software and Protocol Stacks	Bluetooth Smart Host Stack & Profiles SMAC, IEEE 802.15.4 MAC, Thread Stack KSDK, KDS, IAR, RTOS		
Availability (subject to change)	General Availability/Production – Sep/Oct'16		

Thread Ultra Low Power End Device

Kinetis L devices with 32kB RAM can run 802.15.4 MAC/PHY, Thread Network and Application as an End Device when paired with the MCR20A 2.4GHz Transceiver



150kB Flash
25kB RAM

FRDM-KL46Z

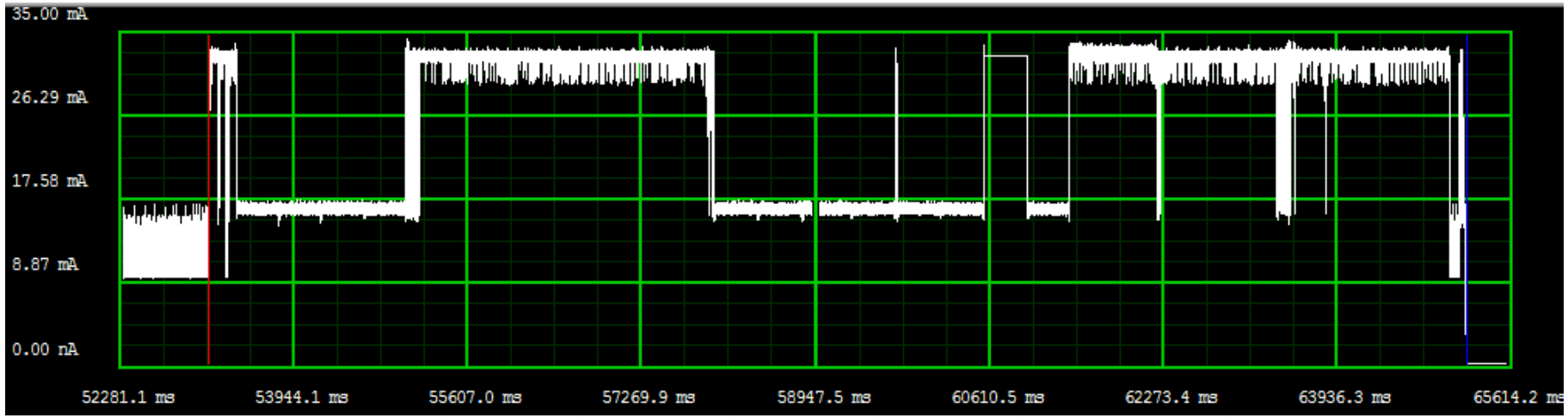
FRDM-CR20A

Example: FRDM KW24D512 – Low Power End Device

- Measured MCU + Radio current
- Clock freq = 32MHz
- TX PWR = 3 (-40dBm)
- Powered from KW24 USB (J16)

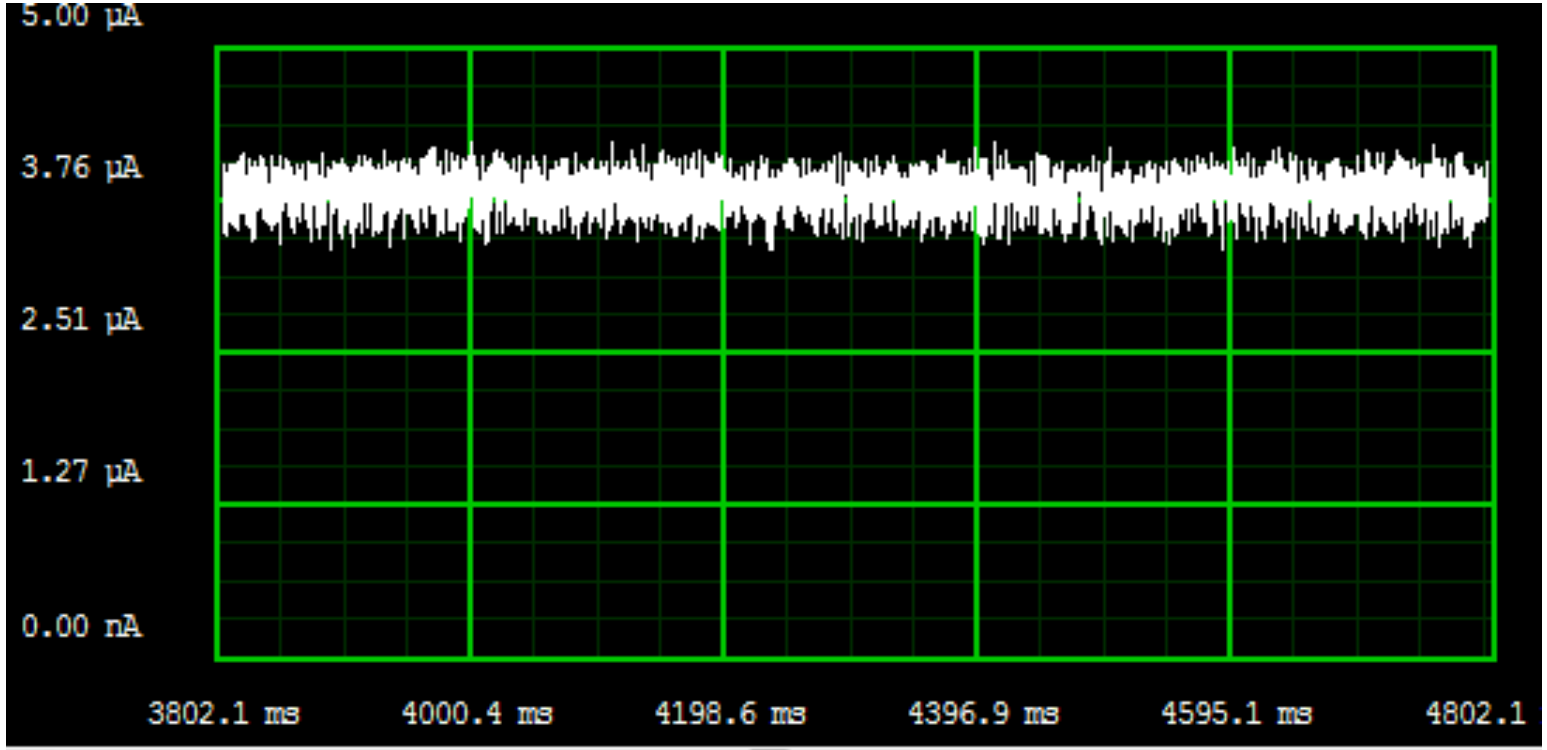
Note: For the polling current calculations an average of 4 polling measurements was taken.

FRDM-KW24D512 - Joining network (Commissioning)



Name	Duration	Min	Max	Average	Standard Dev...	Integral
Probe1/Current Cursor	12.10 s	588.54 µA	34.60 mA	26.20 mA	7.98 mA	317.02 mAs

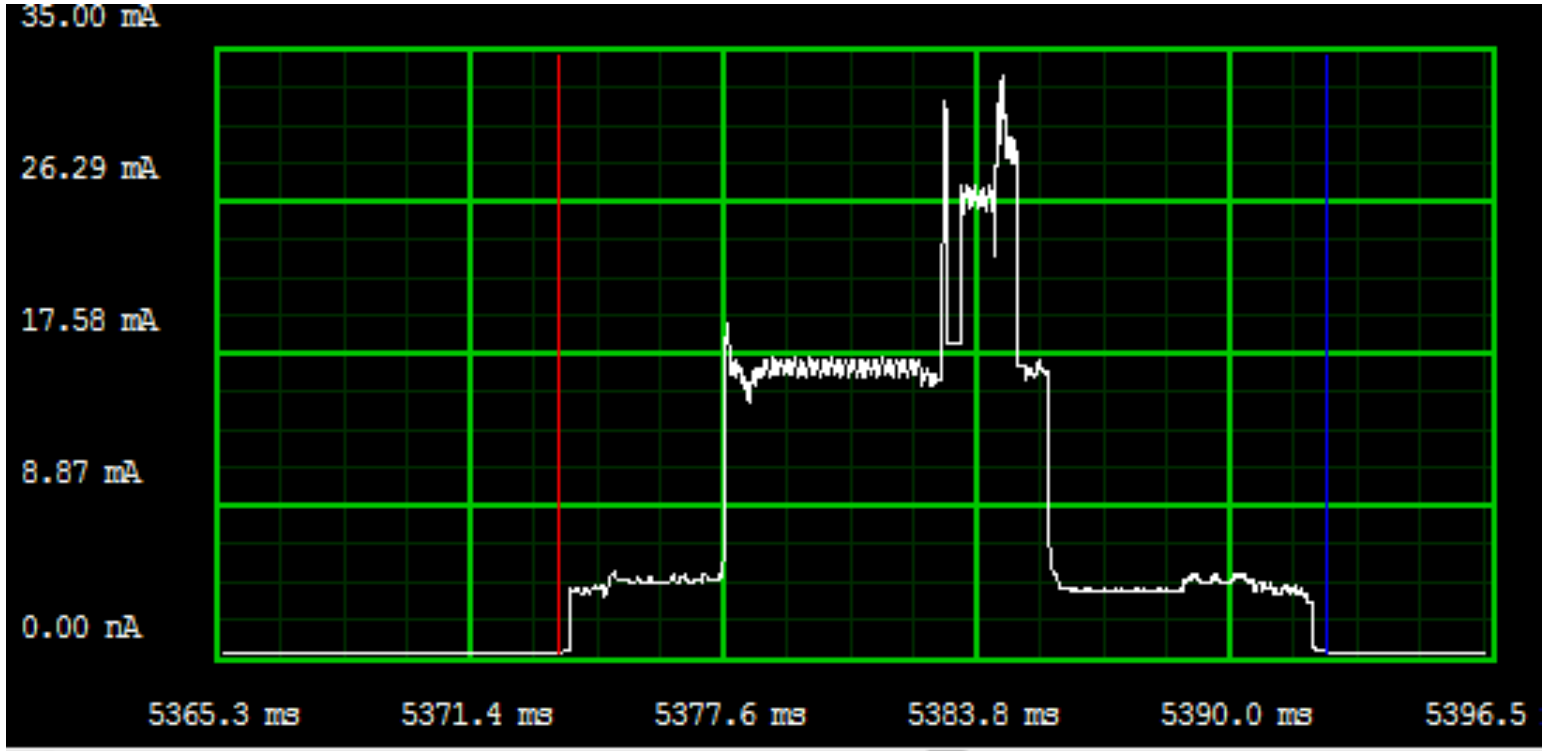
FRDM-KW24D512 - Sleep current



Statistic

Name	Duration	Min	Max	Average	Standard Dev...	Integral
Probe1/Current	1.00 s	3.36 µA	4.26 µA	3.86 µA	128.01 nA	3.86 µAs

FRDM-KW24D512 - Polling current



Statistic

Name	Duration	Min	Max	Average	Standard Dev...	Integral
Probe1/Current Cursor	18.95 ms	35.93 µA	33.73 mA	10.22 mA	8.09 mA	193.68 µAs

FRDM-KW24D512 - Battery life calculation (MCU + Radio)

- Commissioning consumption = 317.02mAs = 0.088mAh
- Time period length = Poll Time + Sleep Time = 20ms + 2980ms = 3000ms
- Average current during poll time: 9.21mA
- Average current during sleep time: 3.86µA

MCU	~ 4uA
Radio	< 1uA
Total	~3.86uA

- Average current on a poll-sleep sequence =
  ((9.21mA * 20ms) + (3.86µA * 2980ms)) / 3000ms = 70.62µA

- With a ½AA LI-SOCI2 3.6V battery with a capacity of 1200mAh the lifetime of the system is:

  (1200mAh - 0.088mAh)/ 70.62µA = 16992 h = 708 days = 1.94 years

KINETIS THREAD STACK CONFIGURATIONS

Thread Network Commissioning Attributes

- **Extended PAN ID (XPANID)** – 8 bytes unique ID, usually quasi-random, uniquely identifies Thread networks in same wireless range
- **Network Name** – similar to Wi-Fi SSID: “Kinetis_Thread”
- **Master Security Key** - 16 bytes used to derive base link layer and mesh link establishment AES 128 encryption / decryption
- **Network Commissioning Password (PSKc)** – needed to authorize external commissioners
  - Are chosen by user or generated randomly, then setup on initial node which creates a network: Router Eligible Device acting as initial Leader Router

Thread Device Commissioning Attributes

- **IEEE EUI64 Address** – 8 bytes consisting in an unique ID (3 byte vendor IEEE OUI
  - e.g. 00:04:9F for Freescale + 5 byte vendor assigned
  - mandatory to be unique for each device
  - **NEVER used as is over the air by Thread** for privacy and security reasons → a SHA-256 hash of the EUI64 is used in commissioning authentication

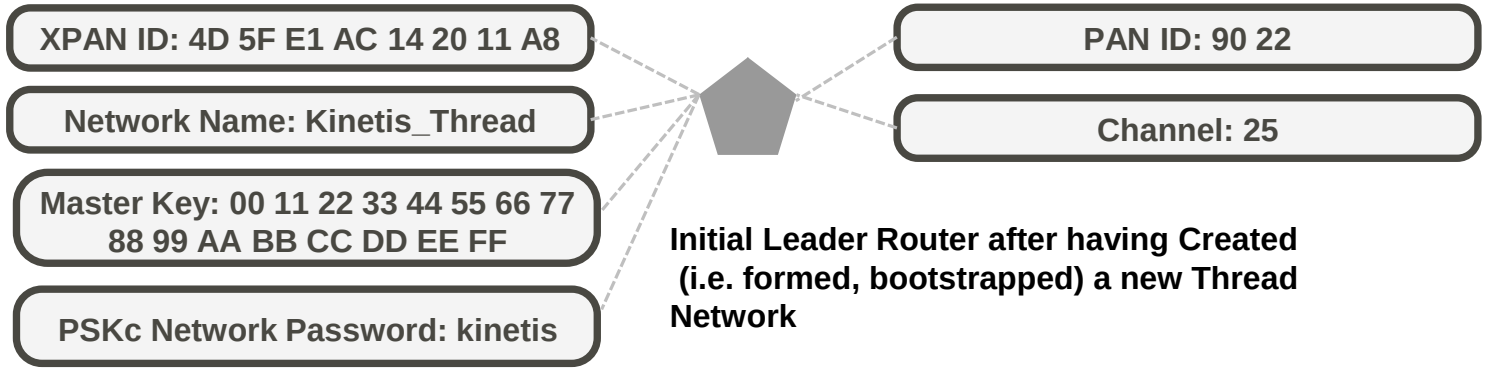
- **Device Password (PSKd)** – set by manufacturer
  - e.g. can be a unique alphanumerical string printed on product label or box

Thread Network Runtime Attributes

- **IEEE 802.15.4 (Short) PANID** – 2 bytes, set at the IEEE 802.15.4 link layer, usually quasi-random uniquely identifies Thread networks in same wireless range
- **IEEE 802.15.4 RF Channel** – in 2.4GHz: 16 channels,

  with IDs 11- 26 (2405 → 2480 MHz)

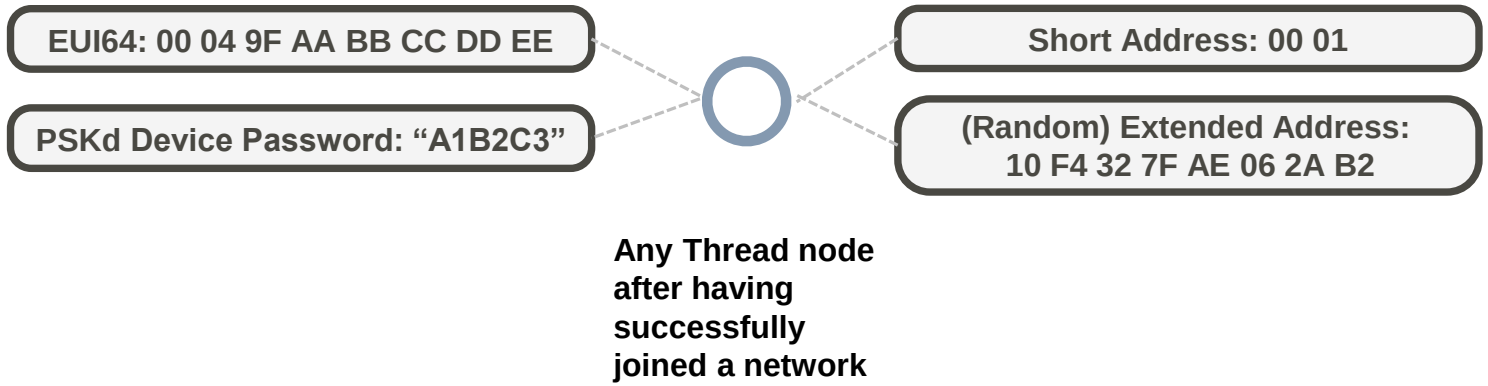
  - initial Leader Router usually selects a random PAN ID and does a Scan to select a good channel with the least activity
  - channel can be also set by application as “channel mask” (subset of allowed channels)



Thread Device Runtime IEEE 802.15.4 Attributes

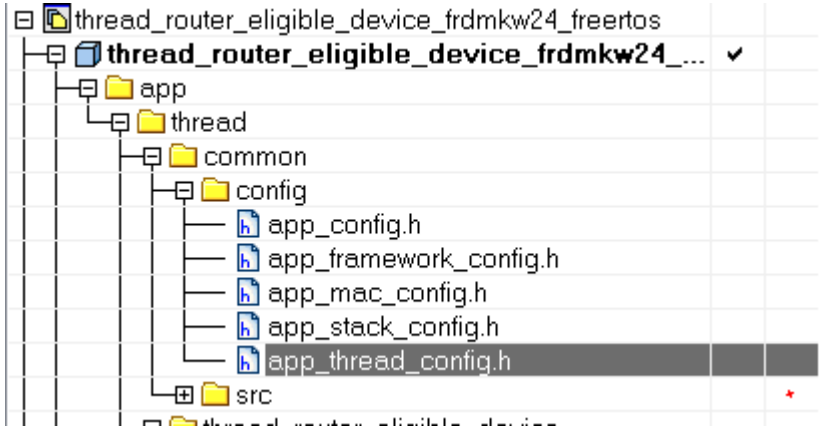
- **IEEE 802.15.4 (Short) Address** – 2 bytes, used and assigned specifically within a RLOC to facilitate locating nodes within the Thread mesh routing algorithm

- **Random IEEE 802.15.4 Extended Address** – 8 bytes randomly generated after commissioning with a TRNG, meant to guarantee device identity in the same network



Hands On – Identify These Settings in the Source Code

- Found in app\thread\common\config\app_thread_config.h



- change channel mask and Network Name to avoid overlaps:

```
#define THR_SCANCHANNEL_MASK   (0x00000001 << CH)
#define THR_NETWORK_NAME                {6, “MYNAME"}
```

- Example:
- CH = 11 for channel 11
- MYNAME = Cristian (make sure you put the correct length for the string)

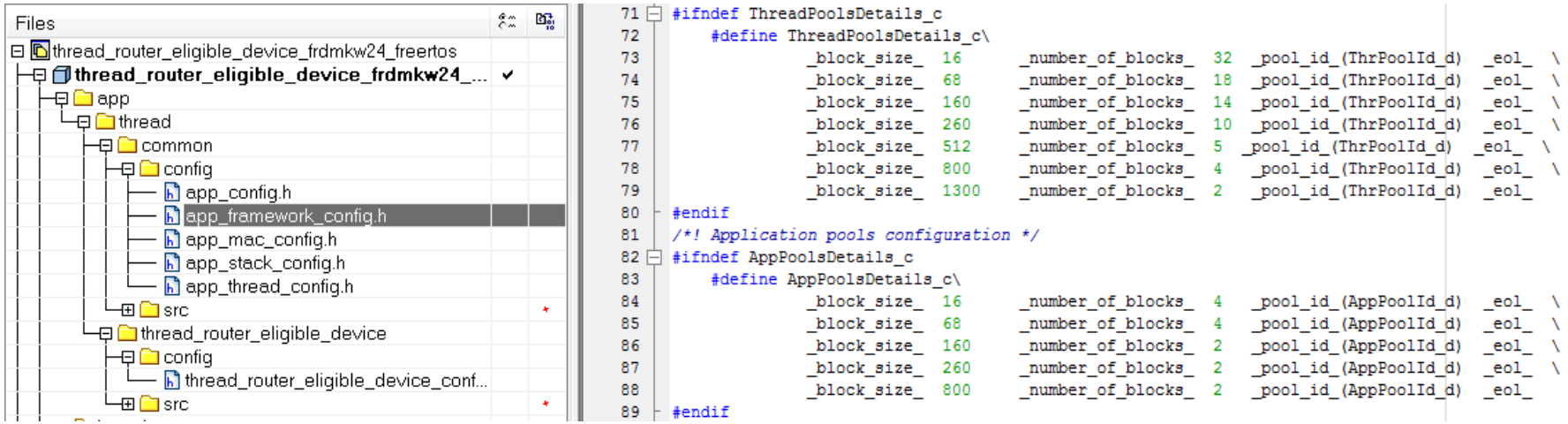
- **Search the code for the other settings:**
  - EUI64, PanID, Extended PanID, Master Key, PSKd

Other Interesting Thread Configurations

```
/*! The device is out-of-band commissioned (the node is pre-configured). This means that the device
 * has all network parameters to directly attach to that network (e.g. master key, PSKc, mesh-local ULA,
 * extended PAN ID, Network name)  */
#ifndef THR_DEV_IS_OUT_OF_BAND_CONFIGURED
   #define THR_DEV_IS_OUT_OF_BAND_CONFIGURED            FALSE
#endif

/*! The device can become an active router. If this capability bit is set FALSE, the device can not promote
 * itself as a router */
#ifndef THR_DEFAULT_CAN_BECOME_ACTIVE_ROUTER
#define THR_DEFAULT_CAN_BECOME_ACTIVE_ROUTER          TRUE
#endif

/*! On Border Router use the below global /64 on-Mesh Prefix  */
#define THR_BR_GLOBAL_ONMESH_PREFIX          { {0xFD, 0x01, 0x00, 0x00, 0x00, 0x00, 0x3E, 0xAD, \
                                                 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00}, \
                                                 64,\
                                               }

/*! Multicast forwarding (MPL) configuration parameters  */
#define THR_STACK_MPL_CFG_ROUTER     .seedLifetime = 3000,          /* ms */ \
                                      .Imin = 50,                /* ms */ \
                                      .Imax = 256,                /* ms */ \
                                      .k = 0x00,                 /* infinite */ \
                                      .nbOfTimerExpirations = 2, \
                                      .useFullInterval = TRUE,   /* full Trickle interval */ \
                                       THR_MPL_CFG_SEED_ID
```

Dynamic Memory Pools Configuration

- Found in app\thread\common\config\app_framework_config.h




```
71  #ifndef ThreadPoolsDetails_c
72      #define ThreadPoolsDetails_c\
73                  _block_size_  16      _number_of_blocks_  32  _pool_id_(ThrPoolId_d)  _eol_  \
74                  _block_size_  68      _number_of_blocks_  18  _pool_id_(ThrPoolId_d)  _eol_  \
75                  _block_size_  160     _number_of_blocks_  14  _pool_id_(ThrPoolId_d)  _eol_  \
76                  _block_size_  260     _number_of_blocks_  10  _pool_id_(ThrPoolId_d)  _eol_  \
77                  _block_size_  512     _number_of_blocks_  5  _pool_id_(ThrPoolId_d)  _eol_  \
78                  _block_size_  800     _number_of_blocks_  4   _pool_id_(ThrPoolId_d)  _eol_  \
79                  _block_size_  1300    _number_of_blocks_  2   _pool_id_(ThrPoolId_d)  _eol_
80  #endif
81  /*! Application pools configuration */
82  #ifndef AppPoolsDetails_c
83      #define AppPoolsDetails_c\
84                  _block_size_  16      _number_of_blocks_  4   _pool_id_(AppPoolId_d)  _eol_  \
85                  _block_size_  68      _number_of_blocks_  4   _pool_id_(AppPoolId_d)  _eol_  \
86                  _block_size_  160     _number_of_blocks_  2   _pool_id_(AppPoolId_d)  _eol_  \
87                  _block_size_  260     _number_of_blocks_  2   _pool_id_(AppPoolId_d)  _eol_  \
88                  _block_size_  800     _number_of_blocks_  2   _pool_id_(AppPoolId_d)  _eol_
89  #endif
```

- Can be used to tune the RAM memory occupation

- **NOTE: Extensive testing should be done when these settings are changed! High risk of breaking overall stack functionality and robustness**

SHELL INTERFACE DESCRIPTION AND USAGE

Kinetis THREAD Connectivity Shell

- The Shell module is offering a command line user interface over a serial interface, usually UART.
- Implements a basic set of commands (not as many as FSCI does).
- The commands can be viewed by typing the ‘help’ command in the host serial terminal application.
- The Shell can be used for several purposes, including configuring the network, testing the network connectivity, MAC filtering, sockets usage and so on.

```
$ help
help                print command description/usage
version             print version of all the registered modules
history             print history

ifconfig            IP Stack interfaces configuration
ping                IP Stack ping IPv4/IPv6 addresses
ping6               IP Stack ping IPv6 only
reboot              MCU Reset
setstackparams      Set Stack Parameters
getnwkparams        Get Network Parameters
getparent           Get parent information
setnwkparams        Set Network Parameters
macfilter           MAC filtering commands
joinnwk             Joins Network
nwkdata             Network data operations
remove              Remove commands
swkeys              Switch Key
setkeys             Set Keys
socket              IP Stack BSD Sockets commands
```

Kinetis THREAD Connectivity Shell

- In order to get help related to a particular command simply type ‘help’ followed by the command name:

```
$ help ifconfig
ifconfig - IP Stack configure IP addresses for the interfaces
   ifconfig all - displays all interfaces and addresses on the device
   ifconfig <interface ID> ip <IP address>
```

- All the shell commands description as well as usage examples are available in THREAD documentation release - THREAD Shell Interface User’s Guide.pdf

How to add more commands to the shell interface

- Add in your app file the definition of the shell command

```
#if THREAD_USE_SHELL
ShellComm_RegisterStatic(shellComm, "largenwktest", 5, 0, SHELL_LargeNwkTest
#if SHELL_USE_HELP
    ,"Large network test command",
    "Large network test command\r\n"
    "   largenwktest start testmode <peakmcast commmand repeat count>\r\n"
    "                   testmode = unicast/multicast/both\r\n"
    "   largenwktest result <Index>\r\n"
    "   largenwktest stop\r\n"
    "   largenwktest reset\r\n"
#endif /* SHELL_USE_HELP */
#if SHELL_USE_AUTO_COMPLETE
    ,NULL
#endif /* SHELL_USE_AUTO_COMPLETE */
   );
#endif
```

Example new shell command

- Add the function and the prototype to process the new command

```
static int8_t SHELL_LargeNwkTest(uint8_t argc, char *argv[])
{
     command_ret_t ret = CMD_RET_ASYNC;

   /* Stop pending commands */
   if(argc == 0)
   {
      shell_refresh();
      ret = CMD_RET_SUCCESS;
      return ret;
   }

   /* lartgenwk start */
   if(!strcmp(argv[1], "start"))
   {
      /* lartgenwk start unicast*/
      if(!strcmp(argv[2], "unicast"))
      {
         testmod = Unicast_only;
      }
   ……….……….
   }

   /* lartgenwk stop */
   else if(!strcmp(argv[1], "stop"))
   {
      if(LargeNwkTestTaskId != 0)
      {
        shell_write("stopping test ....\r\n");
        Largenwk_stop();
      }
```

Compare for certain keywords

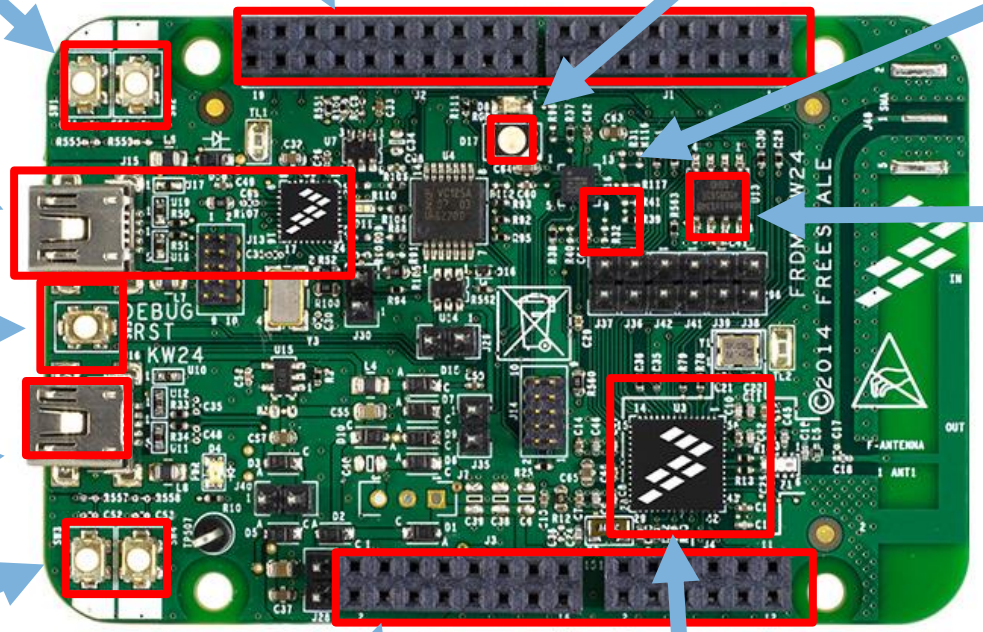
Run the respective command

HANDS-ON: ADDING AN I2C SENSOR TO A KINETIS THREAD ROUTER ELIGIBLE DEVICE

Thread Hands On

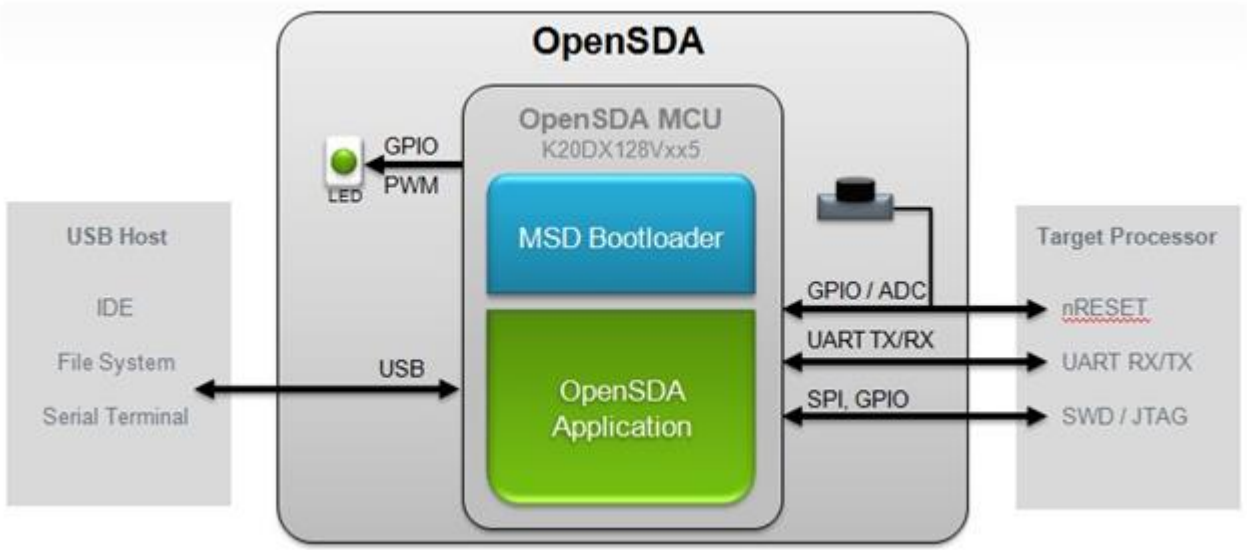
- This hands-on lab will explore a basic Thread network example, show how to add new CoAP commands, show how to use Thread timers, and show how to add a new shell command
- Uses two FRDM-KW24D512 boards
- Lab guides are being handed out and can also be found on the desktop of your computer

FRDM-KW24D512 Hardware Overview



OpenSDA

- OpenSDA is a circuit built into Freescale evaluation boards to provide a bridge between your computer and the embedded target processor
- Purpose is to provide inexpensive debug tool for Freescale evaluation boards
- Different apps can be loaded via a bootloader
- Default CMSIS-DAP app does:
  - Drag-and-drop flashing via a Mass Storage Device
  - Debug via CMSIS-DAP protocol
  - Virtual Serial Port



Hands On with Thread

- **Part 1:** Thread Basics
  - Create simple Thread network with two router eligible devices
  - Get Thread network information via shell commands
  - Explore the out of box demo

- **Part 2:** Add Accelerometer CoAP capabilities
  - Add code to access on-board accelerometer via I2C
  - Add new action for button press to send accelerometer data
  - Create a new CoAP command to send and receive accelerometer data between boards

- **Part 3:** Thread Timers
  - Create timer to send accelerometer data at regular intervals
  - Add new shell command to start and stop the timer

LINKS OF INTEREST

Links of interest

- Kinetis Thread: www.nxp.com/Thread
- Thread Group: www.threadgroup.org

- Kinetis Thread Extranet (download): www.nxp.com/go/ThreadBeta

- KW2x: www.nxp.com/KW2x
- FRDM-KW24D512: www.nxp.com/FRDM-KW24D512
- USB-KW24D512: www.nxp.com/USB-KW24D512

- KW41Z: www.nxp.com/KW41Z
- FRDM-KW41Z: www.nxp.com/FRDM-KW41Z
- USB-KW41Z: www.nxp.com/USB-KWKW41Z

- FRDM-K64F: www.nxp.com/FRDM-K64F
- FRDM-KL46Z: www.nxp.com/FRDM-KL46Z
- FRDM-CR20A: www.nxp.com/FRDM-CR20A

Links of Interest

- NXP Thread Beta Community:
  - https://community.freescale.com/community/wireless-connectivity/thread-beta

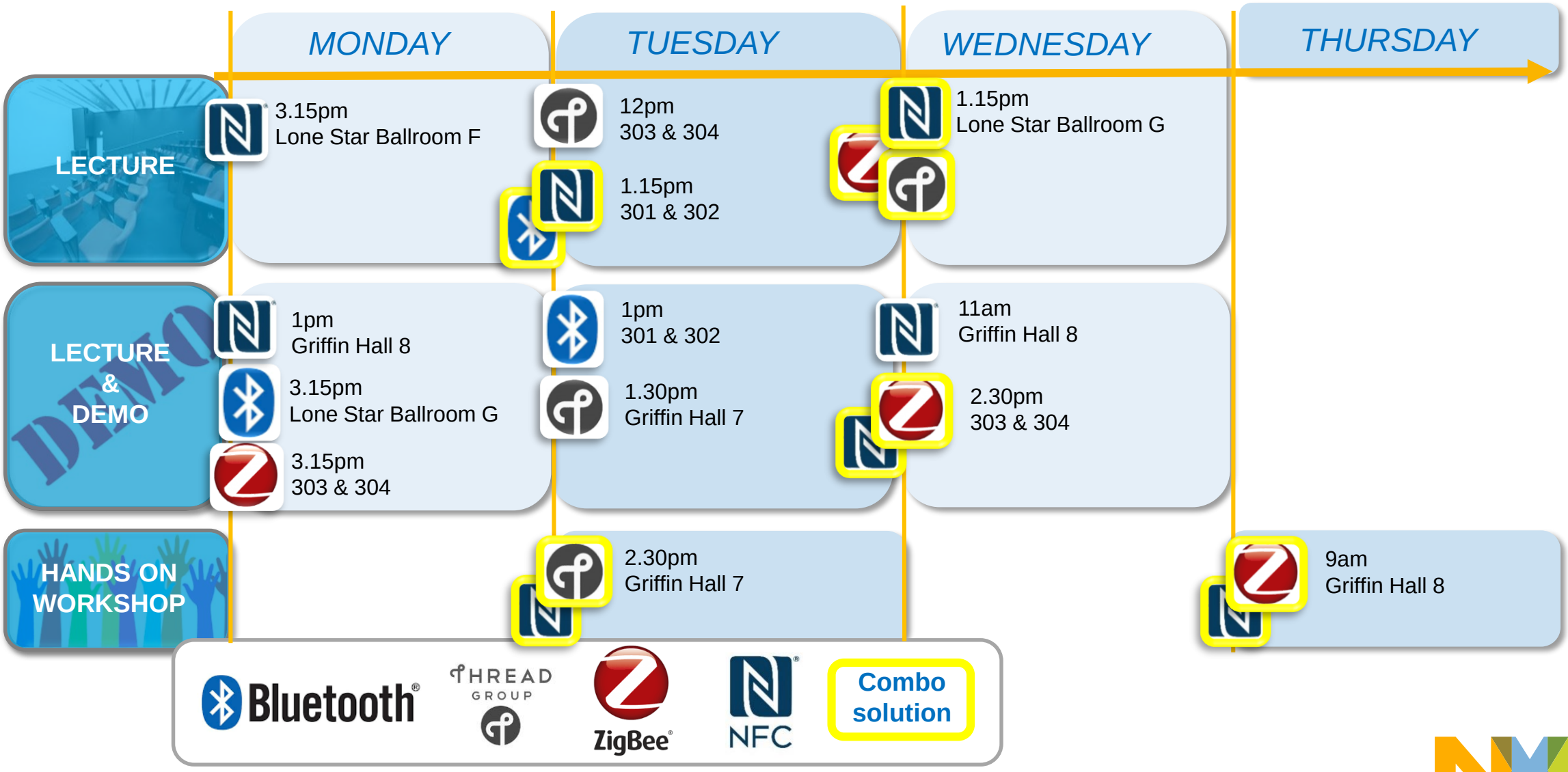
contact product team for access

- NXP Thread Public Community:
  - https://community.freescale.com/community/wireless-connectivity/content?filterID=contentstatus%5Bpublished%5D%7Ecategory%5Bthread%5D

- NXP Wireless Connectivity Community:
  - https://community.freescale.com/community/wireless-connectivity

- Videos:
  - Embedded World 2016 - Thread Intrepid Apps Commissioning demo: https://youtu.be/3bVgv5eCBzs
  - Embedded World 2016 - Thread NFC Commissioning demo: https://youtu.be/X7cVBjMIp04
  - FTF2015
  - http://www.nxp.com/video/thread-smart-home-connectivity-demonstrations:FTF-THREAD-DEMO
  - http://www.nxp.com/video/freescale-thread-demo-with-alljoyn-application-framework:THREAD-DEMO
  - Large Network setup
  - https://www.youtube.com/watch?v=wCCyBIxdJo4

Check additional Smart Home sessions around connectivity



SECURE CONNECTIONS
FOR A SMARTER WORLD

ATTRIBUTION STATEMENT