NXP WIRELESS SOLUTIONS: KW41Z WITH BLE AND THREAD

ANTHONY HUERECA

AMF-CNS-T2642 | JUNE 2017

SECURE CONNECTIONS
FOR A SMARTER WORLD

NXP Value Proposition for the IoT Market

LOW POWER

- Ultra-efficient dynamic power
- Ultra-low static power consumption with full SRAM retention
- Low-power peripherals
- Tools for low power design, e.g. power estimation, power profiler, and consumption calculator

SECURE

- Multiple levels of scalable security for ultimate flexibility and protection
- Ensuring communications, software and physical systems are protected from todays threats

CONNECTIVITY

- State-of-the-art RF performance
- Choice of connectivity to fit application
- Interoperable connectivity
- Integrated RF transceivers supporting Bluetooth Smart, IEEE 802.15.4, Thread and zigbee

USER EXPERIENCE

- 'Tap-N-Pair' NFC Commissioning for best-in-class consumer experience
- Bring voice detection & triggering features to wide range of products

QUICK TO MARKET

- Complete kits simplify design and lower risk – get to final product design quickly
- Full ecosystem including application software and cloud connectivity

APPLICATION-IN-A-BOX

NXP Solutions for IoT

NXP products and solutions for:

- Microcontrollers & Microprocessors
- Connectivity: Bluetooth Smart, 802.15.4, Thread, zigbee
- Near Field Communications (NFC)
- Secure Element

Targeted solutions for:

- Simplified device commissioning
- Secure commissioning and user authentication
- Secure processing/transactions
- Voice recognition and triggering
- Always-on sensor processing
- Interoperable wireless connectivity

NXP Kinetis Wireless Solutions Agenda

- **KW41 Family Overview**

- **Bluetooth Low Energy (BLE)**

- **Thread**

- **BLE+Thread**

- **Wireless Framework**

- **Modular Gateway**

KINETIS KW41Z/31Z/21Z WIRELESS MCU

KW41Z Family – Single Chip Solution for the IoT

✓ **Multi-Protocol Radio** – High performance radio supporting Bluetooth Smart/Bluetooth Low Energy (BLE) v4.2, Generic FSK and IEEE 802.15.4 (Thread) based standards

✓ **Large Memory** – Enough memory to adequately contain desired networking stack(s) with ample room remaining for custom applications

✓ **Low Power** – Low transmit, receive and standby currents that maximizes battery life, including standard coin-cells

✓ **Complete Enablement** – Fully compliant, certified Bluetooth Low Energy, Thread and 802.15.4 MAC/PHY. Support for Generic FSK, BLE Mesh, SMAC, multiple RTOSes, MCUXpresso SDK and MCUXpresso and IAR IDEs.

Kinetis KW41Z/31Z/21Z

Core/System
- Cortex-M0+ running up to 48 MHz
- Four independently programmable DMA controller channels

Memory
- Up to 512kB Flash
- Up to 128 kB SRAM

Radio
- Support for BLE v4.2, 802.15.4, Generic FSK
- -95 dBm in BLE mode, -100 dBm in 802.15.4 mode
- -30 to +3.5 dBm programmable output power
- 6.8 mA Rx & 6.1 mA Tx (0dBm) current target (DC-DC enabled)
- On-chip balun with single ended bidirectional RF port

Communications/HMI/Timers
- 2xSPI, 2xI2C, LP-UART, GPIO with IRQ capability (KBI)
- Carrier Modulated Timer (CMT) for infrared transmissions
- Hardware Capacitive Touch Sensing Interface (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-128 Accelerator and True Random Number Generator
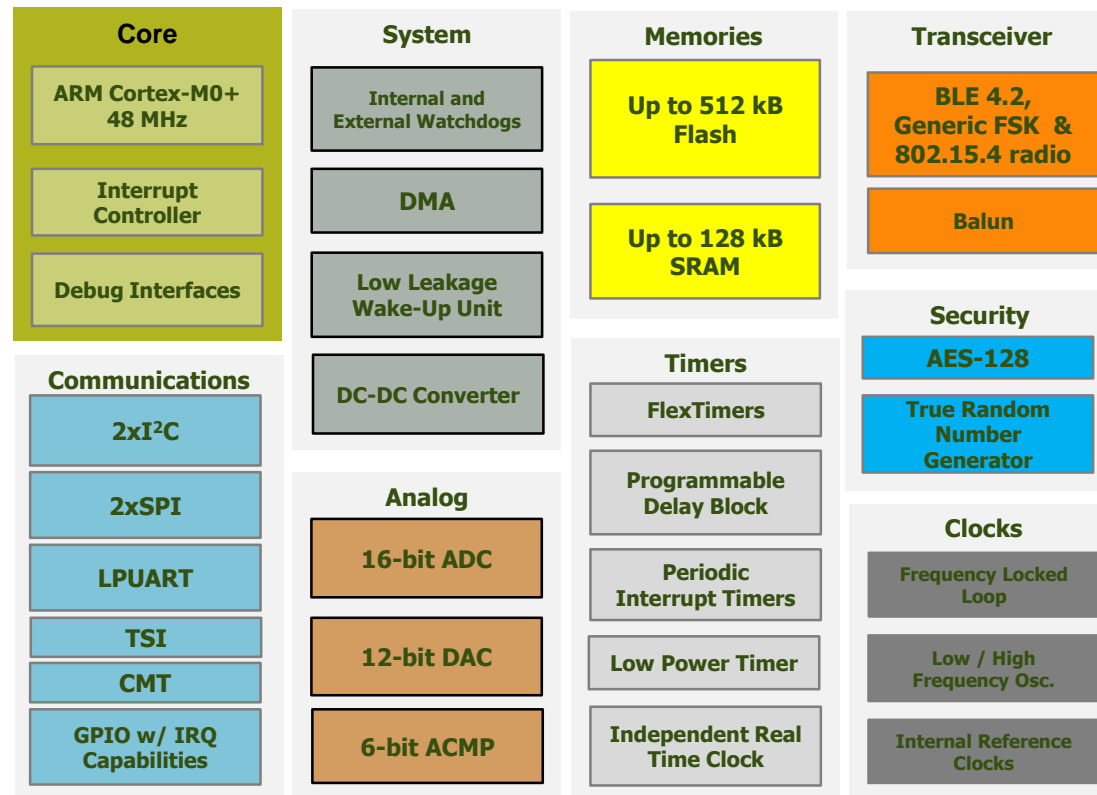- Advanced flash security

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
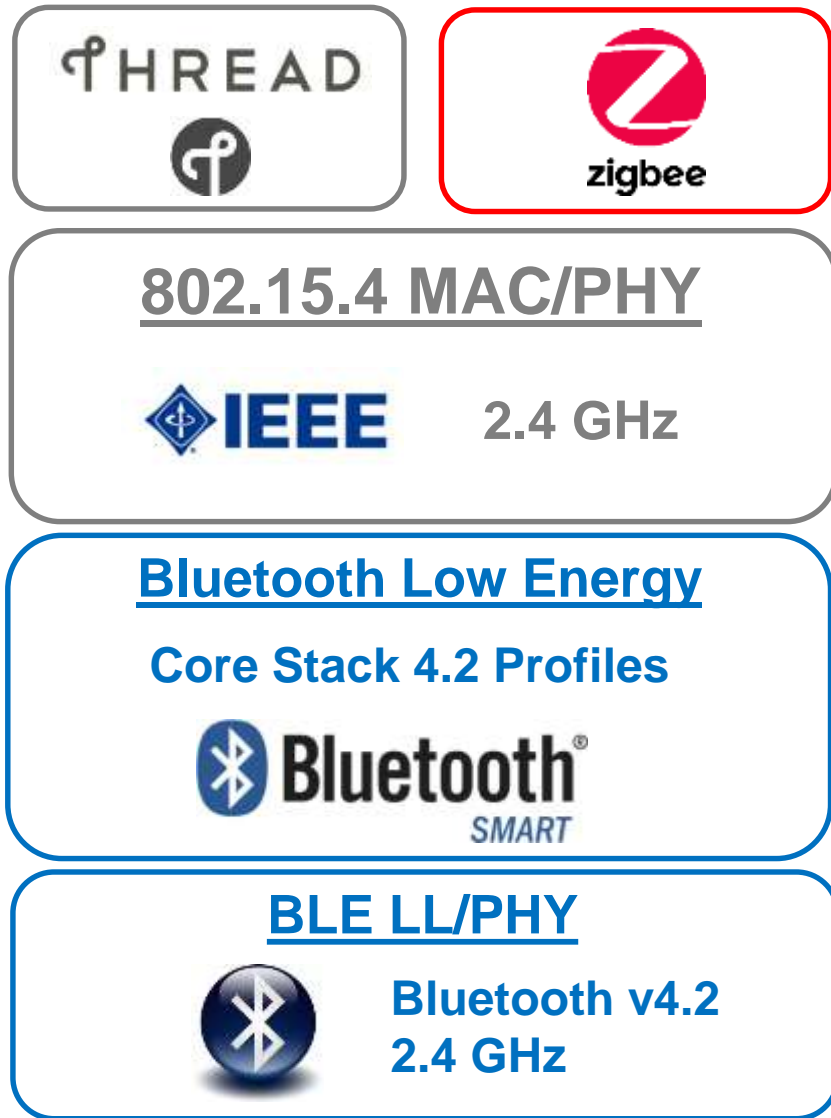
Unique Identifiers
- 80-bit unique device ID programmed at factory
- 40-bit unique media access control (MAC) sub-address can be used for Bluetooth Low Energy or IEEE 802.15.4 MAC Address

-40ºC to +105ºC (QFN), -40C to + 85C (WLCSP)

Core
- ARM Cortex-M0+ 48 MHz
- Interrupt Controller
- Debug Interfaces

System
- Internal and External Watchdogs
- DMA
- Low Leakage Wake-Up Unit
- DC-DC Converter

Memories
- Up to 512 kB Flash
- Up to 128 kB SRAM

Transceiver
- BLE 4.2, Generic FSK & 802.15.4 radio
- Balun

Communications
- 2xI²C
- 2xSPI
- LPUART
- TSI
- CMT
- GPIO w/ IRQ Capabilities

Analog
- 16-bit ADC
- 12-bit DAC
- 6-bit ACMP

Timers
- FlexTimers
- Programmable Delay Block
- Periodic Interrupt Timers
- Low Power Timer
- Independent Real Time Clock

Security
- AES-128
- True Random Number Generator

Clocks
- Frequency Locked Loop
- Low / High Frequency Osc.
- Internal Reference Clocks

Device	Memory (Flash/RAM)	Protocol	Package
MKW21Z512VHT4 MKW21Z256VHT4	512 kB / 128 KB 256 kB / 64 KB	802.15.4	7x7 48-pin Laminate QFN
MKW31Z512VHT4 MKW31Z256VHT4 MKW31Z512CAT4	512 kB / 128 KB 256 kB / 64 KB	BLE v4.2 / Generic FSK	7x7 48-pin Laminate QFN 3.9x3.8 WLCSP (Jun'17)
MKW41Z512VHT4 MKW41Z256VHT4 MKW41Z512CAT4	512 KB / 128 KB 256 KB / 64 KB	BLE v4.2 / Generic FSK / 802.15.4 (Supports concurrent operation)	7x7 48-pin Laminate QFN 3.9x3.8 WLCSP (Jun'17)

Features	Description
Software and Protocol Stacks	Bluetooth Smart Host Stack & Profiles IPv6 over BLE Generic FSK (250 kbps, 500 kbps, 1Mbps) zigbee 3.0 (December) Thread Stack, IEEE 802.15.4 MAC Thread + BLE Multi-Protocol Stack KSDK, RTOSes, IAR & MCUXpresso Support

Complete Enablement: Software

THREAD

zigbee

802.15.4 MAC/PHY

IEEE   2.4 GHz

Bluetooth Low Energy

Core Stack 4.2 Profiles

Bluetooth
SMART

BLE LL/PHY

Bluetooth v4.2
2.4 GHz

- ✓ Qualified Bluetooth Low Energy v4.2 Stack + Application Profiles
- ✓ Thread R1.1 Compliant Network Stack
- ✓ Thread + BLE Combo Stack
- ✓ zigbee 3.0 (December)
- ✓ IEEE 802.15.4 MAC/PHY
- ✓ IPv6 over BLE
- ✓ Generic FSK at 250, 500 and 1000 kbps
- ✓ SMAC w/ Connectivity Test for Regulatory Certification
- ✓ Support for Host MCU and MPU (Linux®) Processors
- ✓ Full integration with MCUXpresso SDK
- ✓ Multiple RTOS, including FreeRTOS and uCOSII (BLE)
- ✓ MCUXpresso IDE
- ✓ IAR Embedded Workbench®

NXP

KW41Z/31Z/21Z Targeted Applications

High Connectivity, Portable Devices, Powered Optimized Applications

Security & Proximity

Healthcare / Fitness

Smart Home
Home and Building Automation

PC Peripherals

Remote Controls

Beacons
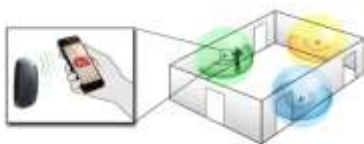
Remote Keyless Entry

WIRELESS SOFTWARE STACKS

BLE

Bluetooth Low Energy Host Stack Facts & Figures

- KW41Z Low Memory BLE Stack footprint
  - **80 - 130 kB flash** and **15 - 30 kB RAM** for BLE Stack, Profile and custom application (including KSDK, RTOS and drivers)
  - **90 kB flash** and **5 kB RAM** for bare metal BLE stack

- Runs on a Cortex™ M0+ @32MHz (20% CPU bandwidth max.)

- Compliant to the Bluetooth® LE v4.2 specification

- The host stack can function with virtually **any** BLE v4.0, v4.1, v4.2 **controller**.

- It is **RTOS agnostic** and can run in a non-preemptive mode (bare-metal). Some loop-based scheduling is still required.

- Coexists with the 802.15.4 MAC and upper network stacks in the same **dual mode firmware** for KW41Z

- Support for **embedded** application development or **Hosted** mode (external MCU/MPU)

- **IAR Embedded Workbench** and **MCUXpresso IDE** support

Enablement: Smartphone App – Kinetis BLE Toolbox

BLE Toolbox include support for the following BLE profiles:

- Glucose
- Blood Pressure
- Cycling Speed and Cadence
- Health Thermometer
- Heart Rate
- Proximity
- Running Speed and Cadence

Also includes Beacon monitoring and support for customs profiles, including:

- Over the Air Programming (OTAP)
- Wireless UART

Also supportsThread + BLE

Download today from iTunes App Store (iOS) or Google Play (Android)

IPv6 over BLE

Bluetooth stack:

- GATT-based profile Internet Protocol Support Profile (IPSP)
- L2CAP Connection Oriented Channels as bearer for 6lo traffic

Adaptation layer:

- 6LoBLE: optimization of IPv6 packets for Bluetooth

IPv6 Stack:

- ICMPv6, UDP, CoAP, ND, etc.
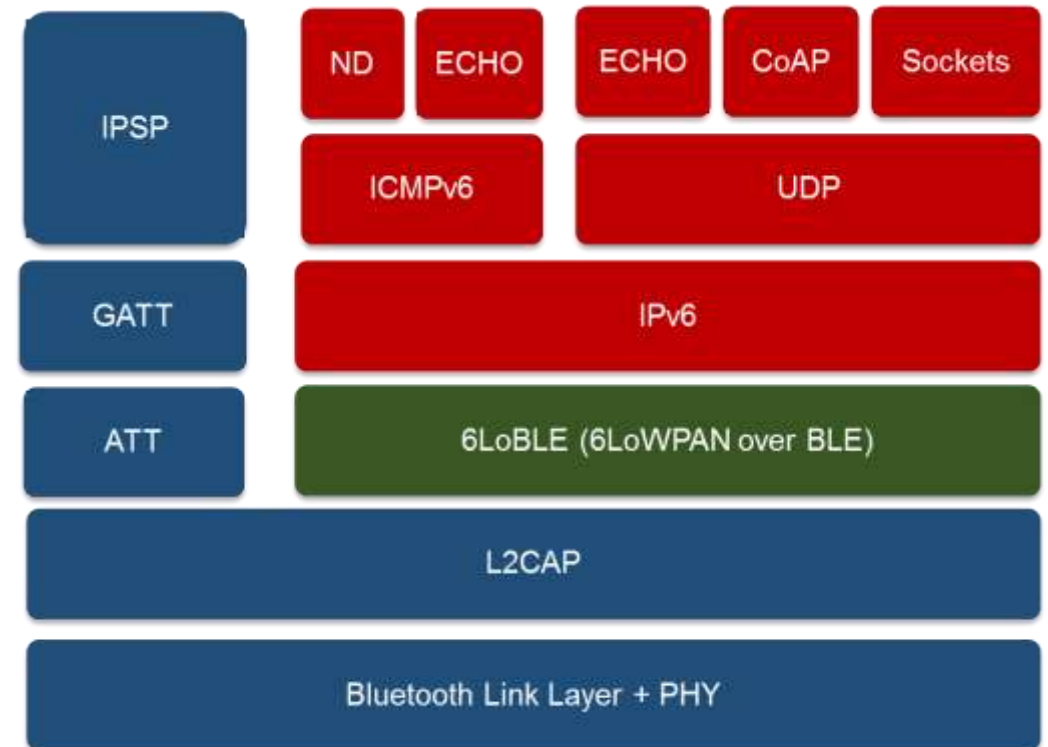
THREAD

⊕HREAD Overview

IPv6 based

Lightweight and low latency

Not a whole new standard

Collection of existing IEEE and IETF standards

Runs on existing 802.15.4 based products

250+ devices on a PAN

- Direct Addressability of devices
- Flexible network with full point to point connectivity of all devices
- No single point of failure
- Enable low cost bridging to other IP networks
- Simple security and commissioning
- Low Power support for sleeping devices

⊕HREAD   Standard

⊕HREAD	Standard
Application Layer	
UDP + DTLS	RFC 768, RFC 6347, RFC 4279, RFC 4492v RFC 3315, 5007
Distance Vector Routing	RFC 1058, RFC 2080
6LowPAN (IPv6)	RFC 4944, RFC 4862, RFC 6775
IEEE 802.15.4 MAC (including MAC security)	IEEE 802.15.4 (2006)
IEEE 802.15.4 PHY	IEEE 802.15.4 (2006)

NXP
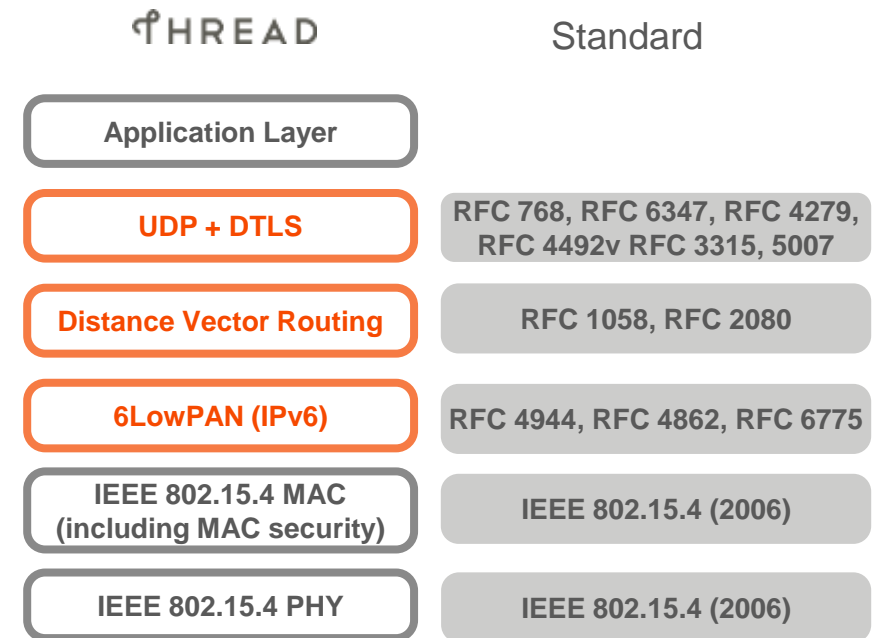
Thread

- 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
  - Includes mandatory security architecture
  - Simple and secure to add / remove products
  - Scalable to 250+ products per network
  - Doesn't require dedicated repeaters
  - Designed for very low power operation
  - Overcomes wireless interference

Can support many popular
application layer protocols and platforms
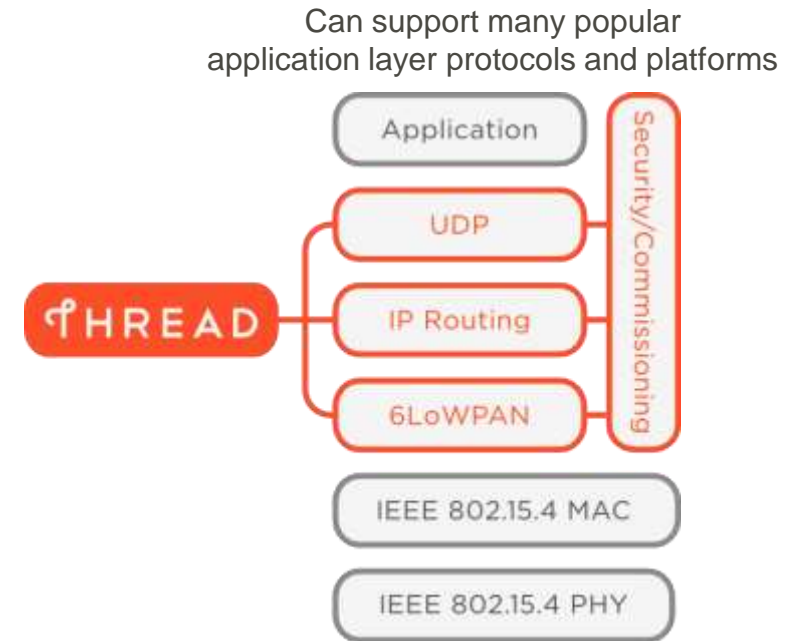
A software upgrade can add Thread to
currently shipping 802.15.4 products

Thread Specification is available to Thread Group members

THREAD + BLE

Multi-Protocol Application

Door Lock using Thread and Bluetooth Low Energy

- **Smart Door Lock contains KW41Z Multi-Protocol Radio**
  - Bluetooth Low Energy
  - Thread (802.15.4)

- **Direct and Network Controlled and Monitoring**
  - Out-of-band commissioning of device on Thread network using BLE
  - Control directly from BLE enabled phone
  - Control and monitoring using cloud connected Thread mesh network
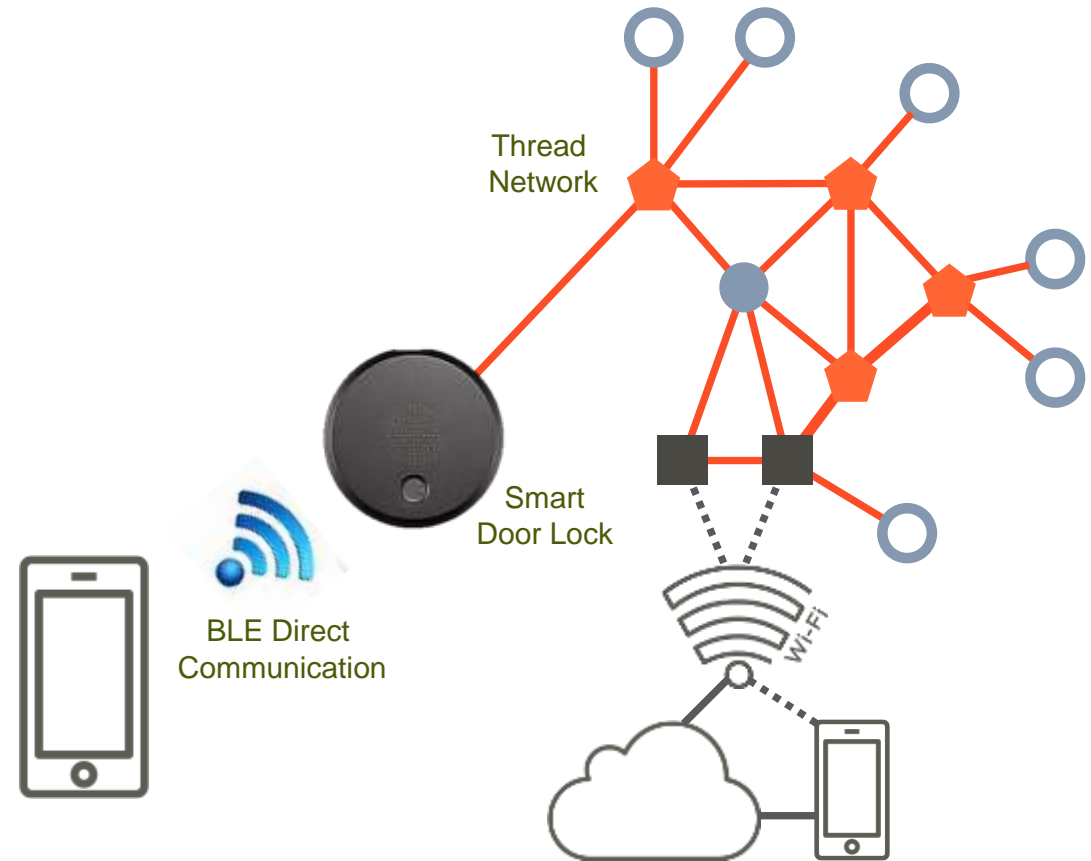
Thread Network

Smart Door Lock

BLE Direct Communication

Wi-Fi

Thread + BLE KW41 Demo Application

BLE UART to Thread Shell

BLE Direct Communication

FRDM-KW41Z

Thread Network
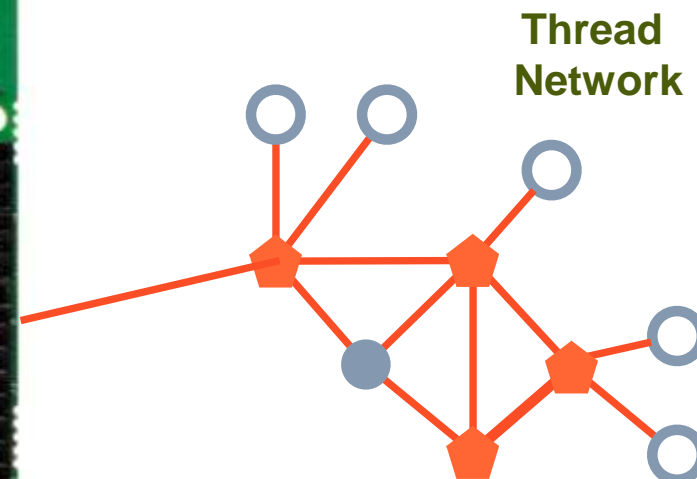
KW41Z ENABLEMENT

KW41Z Development Hardware

- **FRDM-KW41Z** Freedom Development Hardware
  - Can be configured as Host or Shield for connection to Host Processor
  - Supports all DC-DC configurations
  - PCB inverted F-type antenna
  - Minimum number of matching components
  - FCC Part15 & EN300 328 compliant
  - Serial Flash for OTA firmware upgrades
  - On board NXP FXOS8700CQ digital sensor, 3D Accelerometer ($\pm2g/\pm4g/\pm8g$) + 3D Magnetometer
  - OpenSDA and JTAG debug
  - Full KSDK support
  - Resale $145 (2 boards/kit)

- **USB-KW41Z** USB Dongle
  - Ideal for BLE/802.15.4 sniffer or connection to PC/Tablet
  - FCC Part15 & EN300 328 compliant
  - Resale $60

FRDM-KW41Z Hardware Overview
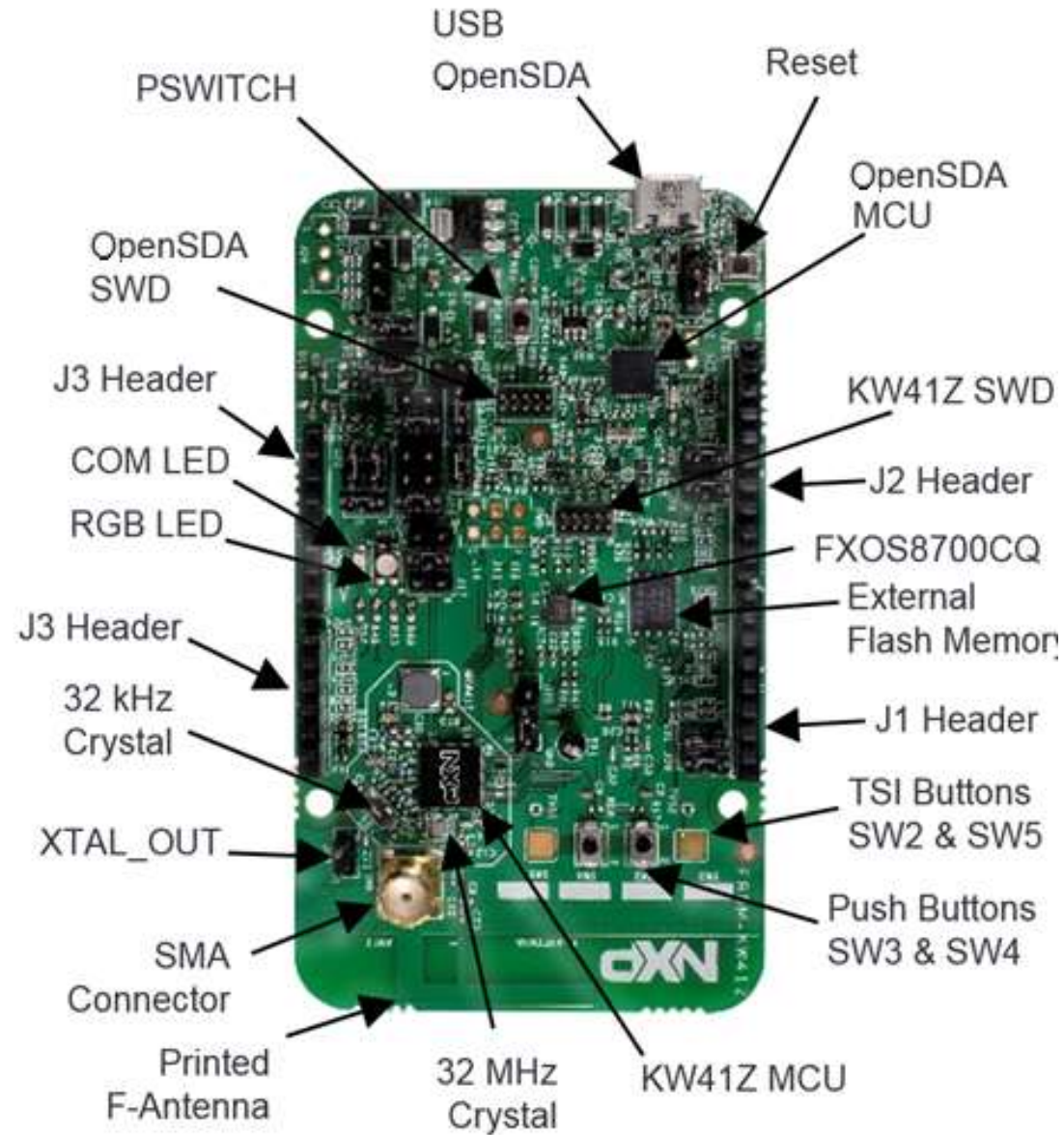
Kinetis KW41Z Connectivity Modules

Rigado
KW41Z512
Arrow

Weptech
KW41Z512
Arrow
EBV

Volansys
KW41Z512

Lierda
KW41Z512
KW31Z256

SMK Japan
KW21Z512

SMK US
KW40Z160
KW41Z512

Not Pictured: Accton (Taiwan), Argenox (AMR), Technexion (Taiwan/Avnet)

Note: See Kinetis W Connectivity Module deck for more details

Introducing …

MCUXpresso Software and Tools

for Kinetis and LPC microcontrollers

MCUXpresso IDE

Edit, compile, debug and optimize in an intuitive and powerful IDE

MCUXpresso SDK

Runtime software including peripheral drivers, middleware, RTOS, demos and more

MCUXpresso Config Tools

Online and desktop tool suite for system configuration and optimization

Summary: Kinetis KW41Z/31Z/21Z

A **true single chip** wireless MCU solution for the IoT

Multi-Protocol Radio
- Integrated, high performance radio capable of running two stacks concurrently saving board space, development time and cost.

Large Memory Footprint
- Large Flash and RAM capacity for running networking stack(s), application profiles and customer application.
- Future proof design with over-the-air upgrade support

Low Power
- Low power design to support products that can run for years on small batteries
- Integrated DC/DC converter with buck and boost support

Comprehensive enablement
Certified software stacks, support for professional and complimentary software development tools, development hardware and reference designs for simpler system design, keeping the BOM cost low, system complexity low. and a fast time to market

www.nxp.com/kinetis/wseries

NXP Kinetis Wireless Solutions Agenda

- **KW41 Family Overview**
- **Bluetooth Low Energy (BLE)**
- **Thread**
- **BLE+Thread**
- **Wireless Framework**
- **Modular Gateway**

BLE

Bluetooth®, SMART, SMART READY?

If a products bears this logo…	It's compatible with products bearing any of these logos…		
Bluetooth SMART READY	Bluetooth SMART READY	Bluetooth	Bluetooth SMART
Bluetooth	Bluetooth SMART READY	Bluetooth	
Bluetooth SMART	Bluetooth SMART READY	Bluetooth SMART	

Bluetooth = Bluetooth Basic Rate/Enhanced Data Rate

Bluetooth Smart = Bluetooth Single Mode = Bluetooth Low Energy

Bluetooth Smart Ready = Bluetooth Dual Mode = Bluetooth Basic Rate/Enhanced Data Rate + BLE

Bluetooth and Bluetooth Low Energy

Bluetooth® "Classic"

Master to client networks (headset to handset, etc)

Low latency, moderate data rates (hundreds kbps)

Good for audio from low-rate voice to streaming music

In billions of handsets all around the world

Bluetooth® Low Energy

Master to client networks (sensing device to handset)

Optimized for "sleepier" sensor devices

▪ **Improved battery life** 10-20x over Bluetooth classic

When connected, low latency, low- to moderate data rates.

Available in latest handsets

Bluetooth LE v4.2 Features

LE Secure Connections

Alignment of security levels and features between BT classic & LE

Key Exchange using Elliptic Curve Diffie-Hellman (ECDH)

LE Privacy v1.2

Controller based Resolvable Private Addresses (RPAs)

LE Data Length Extension

Longer payload lengths – up to 244 octets of application payload

IP Connectivity

IPSP Internet Protocol Support Profile

BLE ARCHITECTURE

BLE System Architecture

Generic Access Profiles (GAP) – what we can do…

Generic Attribute Profile (GATT) – how things are organized

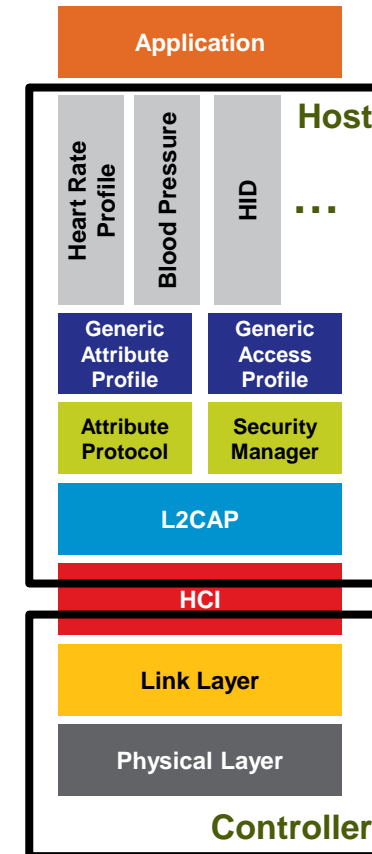**Attribute Protocol (ATT) –** protocol for accessing data

Security Manager (SM) – secures data

Logical Link Control and Adaptation Protocol (L2CAP) – multiplex logical to physical links

Host Controller Interface (HCI) – interface between Host and Controller
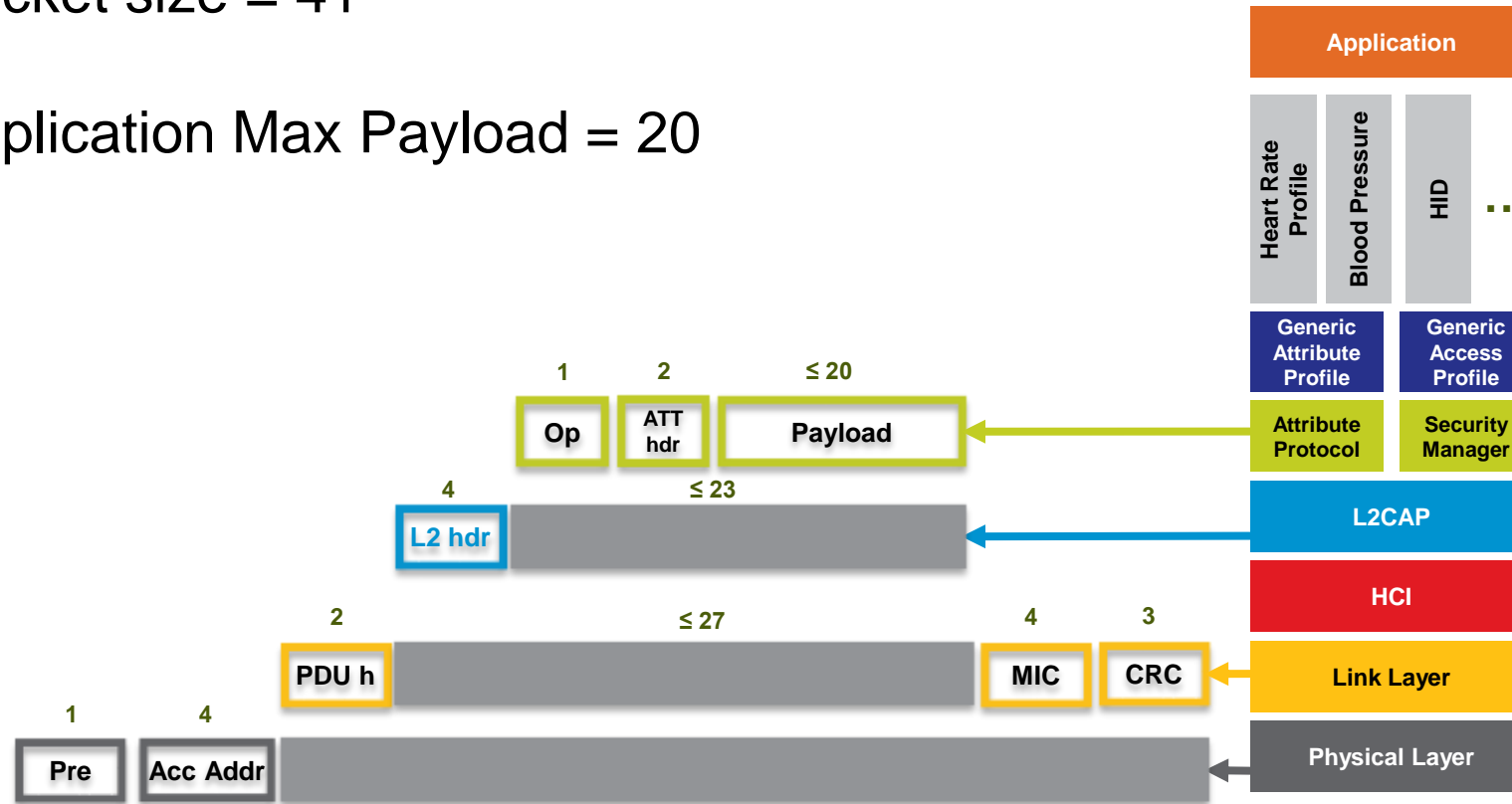
Link Layer (LL) – packets and control

Physical Layer (PHY) – transmits/receives bits
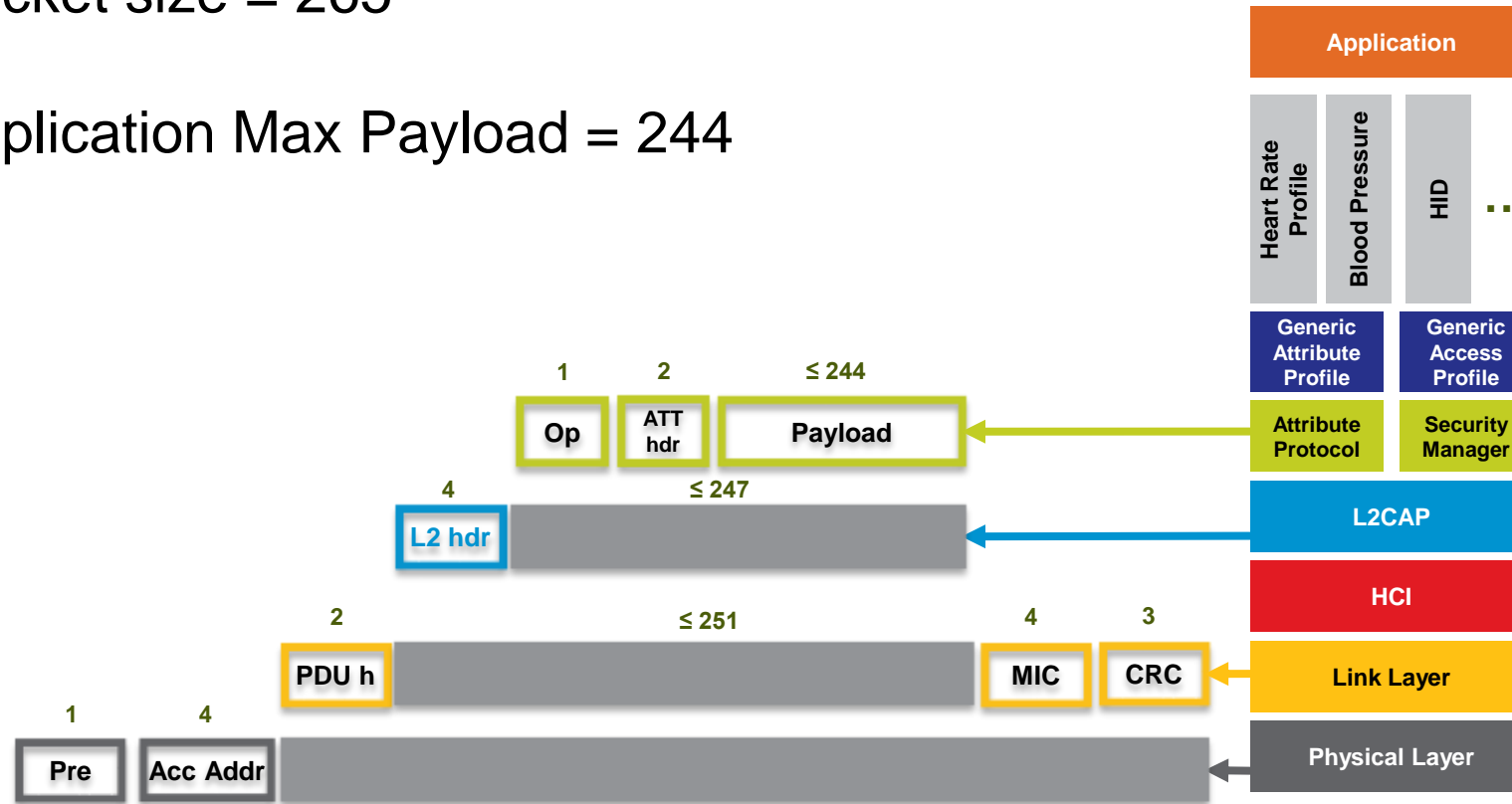
BLE Data Packet
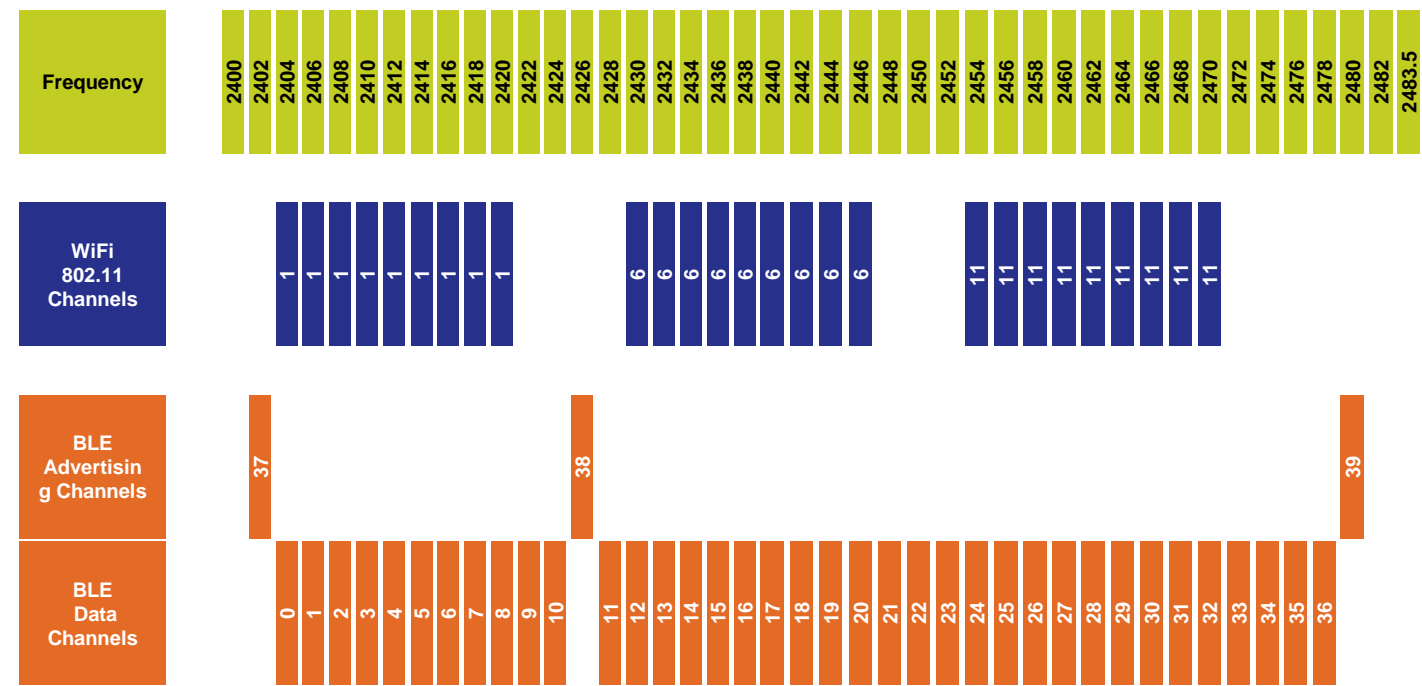
Packet size = 41

Application Max Payload = 20

BLE Max Data Packet

Packet size = 265

Application Max Payload = 244

BLE Channel Assignment

BLE Link Layer

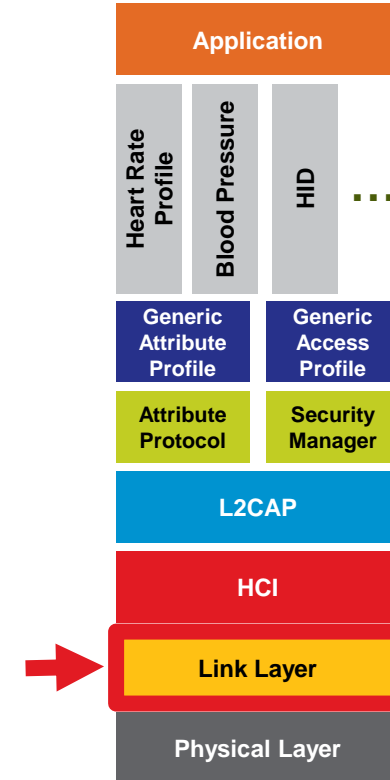Define all connection procedures

Define the frequency hopping mechanism

Manages the timing events during connection

Determine when a connection is lost

Bit Stream Processing
- CRC, whitening, encryption

Application			
Heart Rate Profile	Blood Pressure	HID	…
Generic Attribute Profile		Generic Access Profile	
Attribute Protocol		Security Manager	
L2CAP			
HCI			
Link Layer			
Physical Layer			

BLE Link Layer States

Standby State
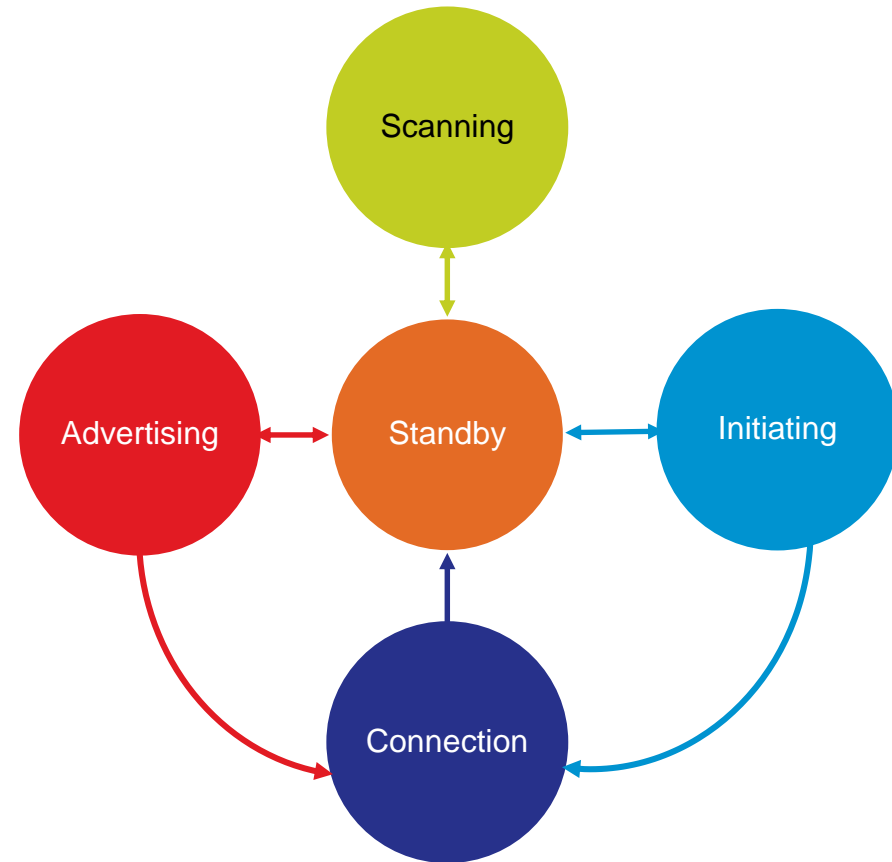Does not transmit or receive packets

Advertising State
Transmitting advertising channel packets
Known as an "Advertiser"

Scanning State
Listening for Advertisers
Known as a "Scanner"

Initiating State
Initiates Connection to the Advertiser
Known as "Initiator"

BLE Link Layer States

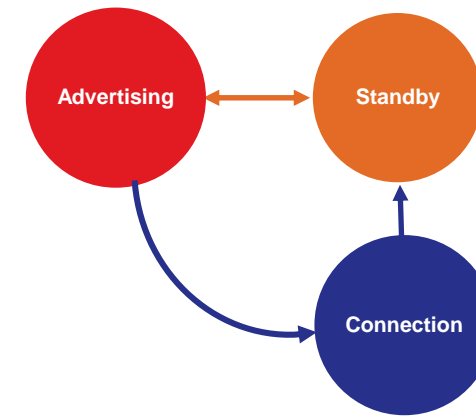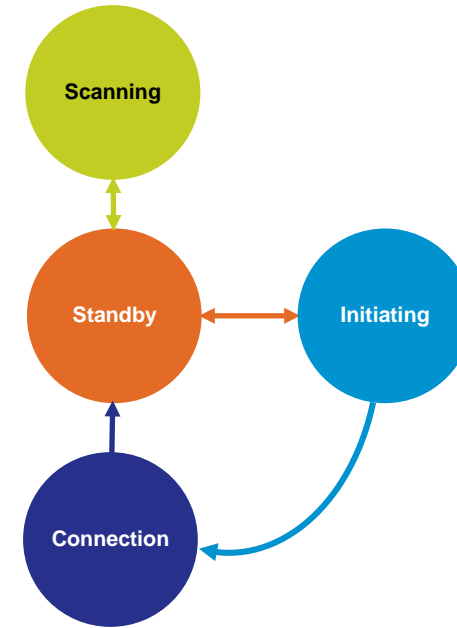## Connection State

Master Role

Entered from Initiating State

Communicates with a device in Slave Role

and defines the timings of transmissions

Slave Role

Entered from Advertising State

Communicates with a single device

in the Master Role

Advertising

Packets meant to discover slaves and connect to them or to broadcast data and connection is not required

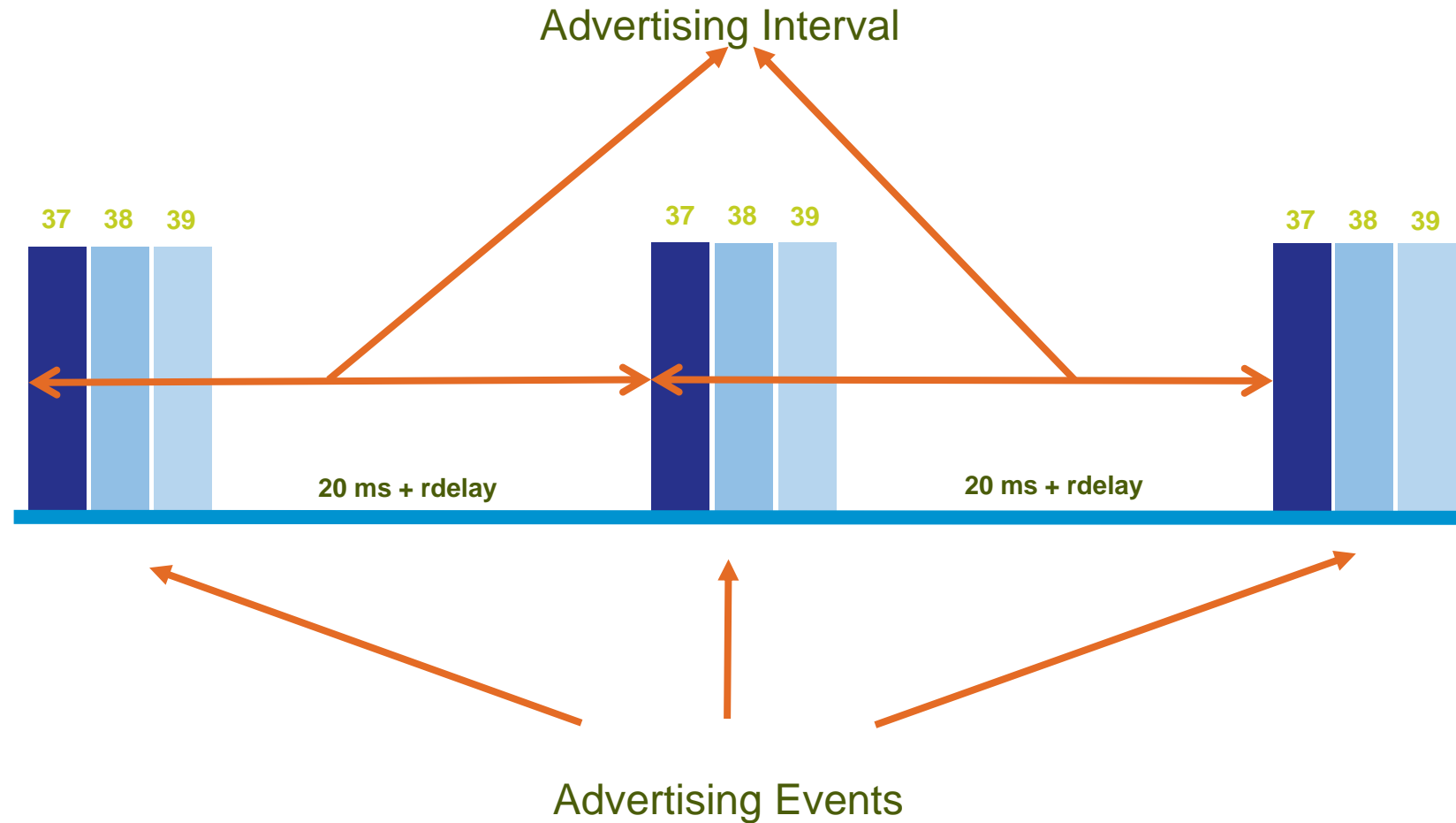Each packet can carry up to 31 bytes of payload
  Useful to filter devices when scanning

These packets are sent in a broadcast at a fixed interval on the 3 advertising channels

Advertising interval can be from 20ms to 10.24 s in 0.625ms steps

Advertising Interval

ADV Interval = 20 ms

Beacon

Non-connectable device that broadcast packets that include identifying information via advertising packets

There are several Beacon protocols:
  iBeacon
  AltBeacon
  Eddystone

The packet structure depends on the protocol used

Coupled with a Smartphone application which reacts to each beacon

Uses **ADV_NONCONN_IND** ADV PDU

Connection

When the master finds a suitable slave, issues a connection request (CONN_REQ)

During the connection request, the master establishes specific parameters to be used during the connection:
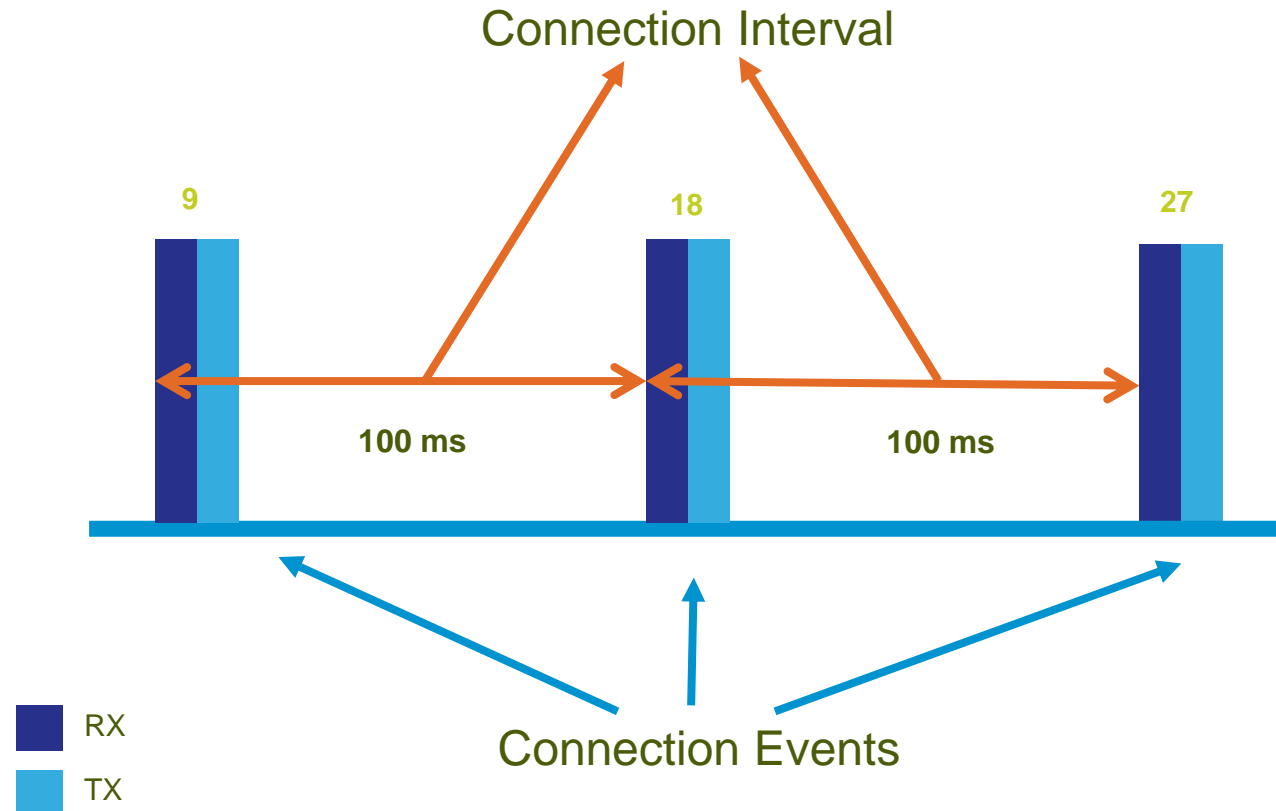 Connection Interval
  Slave latency
  Connection supervision timeout
  Hop Increment

The slave might request an update to these parameters but the master will decide if use the new parameters or keep the previous

Connection (2)

Conn Interval = 100 ms
Hop Increment = 9

Calculating Theoretical Throughput

Connection Interval = 7.5

Packets per connection = 6

Bytes sent per packet = 20

$$AppThroughput = \frac{1000ms * PacketsPerConn * BytesPerPacket}{ConnInterval}$$

Given this formula, the expected throughput is: ~128 Kbps

Example taken from the book "Getting Started with Bluetooth Low Energy" by Kevin Townsend

BLE Host Controller Interface (HCI)

Provides a uniform command interface between the Host and the Controller

Reuse existing Bluetooth HCI Interfaces and transports
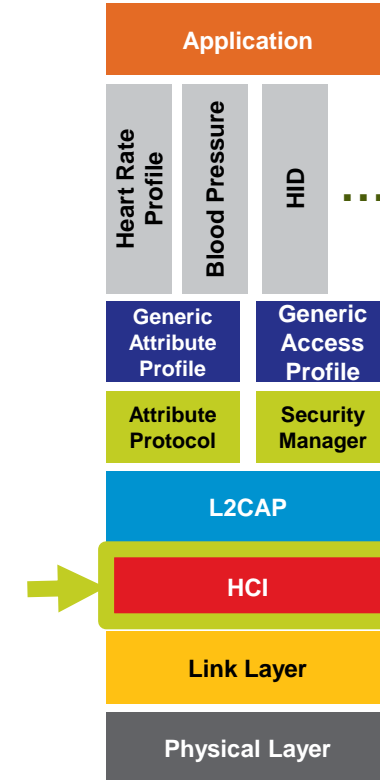
Keeps existing HCI packet formats

BLE commands added for new functionality

Support for Scanning / Advertising Modes

Example commands

- LE Encrypt Command
- LE Rand Command
- LE Receiver Test Command
- LE Transmitter Test Command

Application

Heart Rate Profile	Blood Pressure	HID	...

Generic Attribute Profile	Generic Access Profile

Attribute Protocol	Security Manager

L2CAP

HCI

Link Layer

Physical Layer

BLE Logical Link Control and Adaptation Protocol (L2CAP)

Connection-oriented and connectionless data services to upper layer protocols

Permits higher level protocols and applications to transmit and receive **upper layer data packets**

Similar in function and features to Bluetooth Classic L2CAP
  Reuses existing L2CAP packet format

Fixed Channel IDs (CID)
  CID – local name representing a logical channel end-point on the device
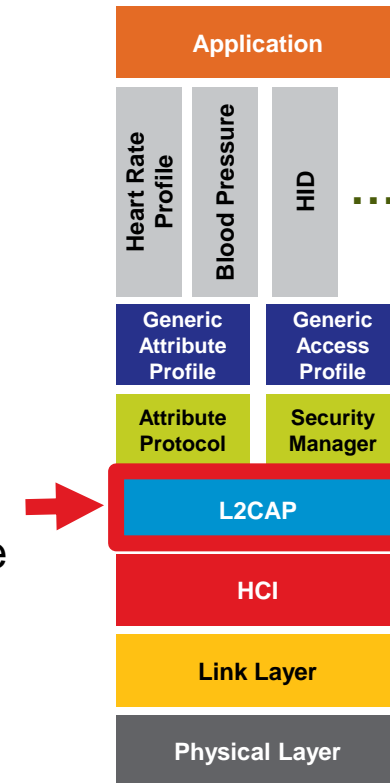 Reduces the traffic between devices thus saving power

L2CAP functions include:
  Protocol/channel multiplexing
  Segmentation and reassembly (SAR)
  Per-channel flow control
  Error control and retransmissions

| Application |
Heart Rate Profile	Blood Pressure	HID	...
Generic Attribute Profile	Generic Access Profile
Attribute Protocol	Security Manager
L2CAP
HCI
Link Layer
Physical Layer

BLE Generic Access Profile (GAP)

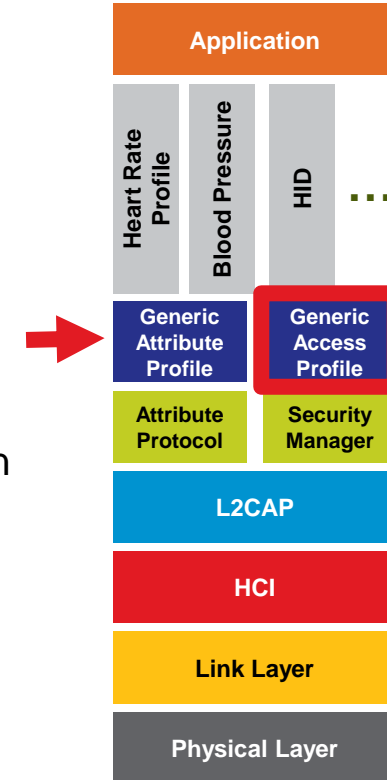## Discoverability modes and procedures

- Allows a device to be discovered by another device
  - Non-discoverable – Cannot be discovered by any device
  - Limited Discoverable Mode – Discoverable for a limited time period
  - General Discoverable – Discoverable for a long time period

Connection modes and procedures

- Allows a device to make a connection with another device
  - Non-connectable – device does not allow a connection to be established
  - Directed Connectable – device accepts a connection request from a known peer device
  - Undirected Connectable – device accepts a connection request from any device

Security/Bonding modes and procedures

- Allows two connected devices to exchange and store security and identity information
  - Non-Bondable – device does not allow a bond to be created with a peer device
  - Bondable – device allows a bond to be created with a peer device

Application
Heart Rate Profile / Blood Pressure / HID ...
Generic Attribute Profile / Generic Access Profile
Attribute Protocol / Security Manager
L2CAP
HCI
Link Layer
Physical Layer

BLE Generic Access Profile (GAP)

Profile roles

- Broadcaster – Sends connectionless data in Advertising Events

- Observer – Receives connectionless data in Advertising Events

- Peripheral – Device that accepts establishment of LE physical link

- Central – Device that initiates the establishment of a physical connection.
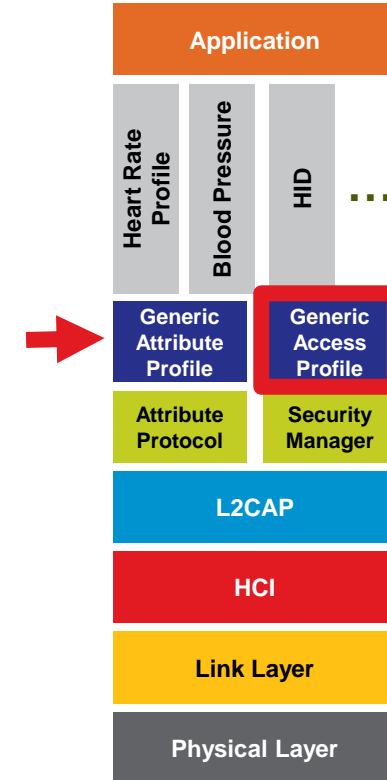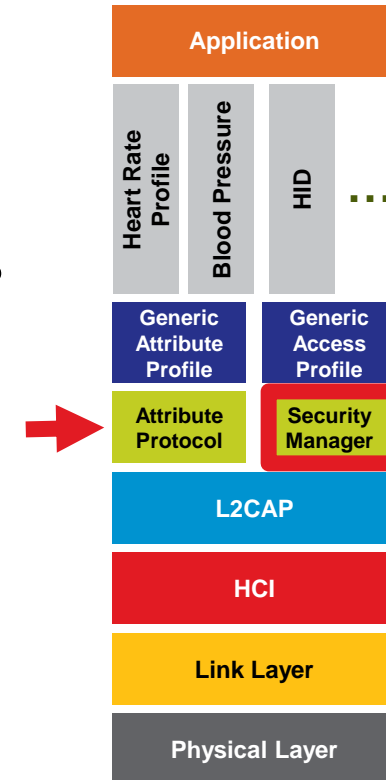
BLE Security Manager

Performs the paring procedure and the link encryption procedure

Security level depends on the device IO capabilities and OOB data exchange capabilities

Long Term Key is calculated, distributed and stored for future connections

LTK is not distributed by the device when LE Secure Connections is used

BLE Security (2)

There are 4 methods for key generation that relates directly on how secure the connection will be:

Legacy Just Works: Key is 0

Legacy Passkey: The key can be from 0 to 999999

Out of the band: Exchange key via NFC, Thread or any other protocol

LE Secure Connection: Devices calculate the key using Elliptic Curve Diffie-Hellman protocol.

   Just Works, Numeric Comparison, Passkey or OOB for authentication

BLE Attribute Protocol (ATT)

Allows a device referred to as a **Server** to expose a set of attributes and their associated values to a peer device referred to as a **Client**

Attributes exposed by **Server** can be discovered, read, written by a **Client**, and can be indicated and notified by the **Server**

Attributes are the smallest data entity defined by the ATT
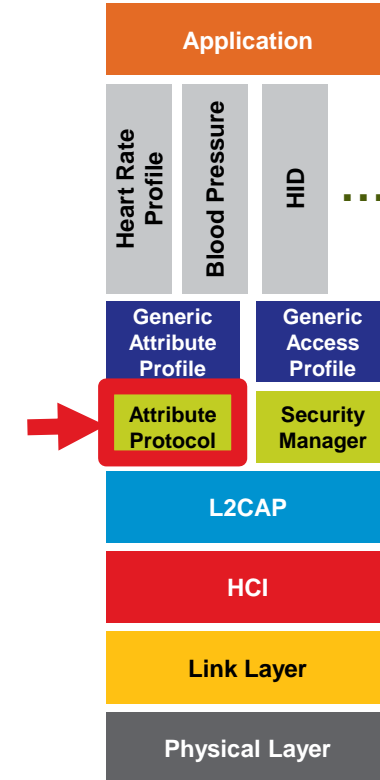
BLE Attribute Protocol (ATT)

Attribute Type

A universally unique identifier (UUID) is used to identify every attribute type

What the attribute represents so the client can understand the attributes exposed by the server

128-bit value, may be shortened to 16-bits

Attribute Handle

Used for accessing the attribute on a server

Attribute Value

Data contained by the attribute

It can represent a measurement value, unit value(km, hours, inches, etc) and information about a device.
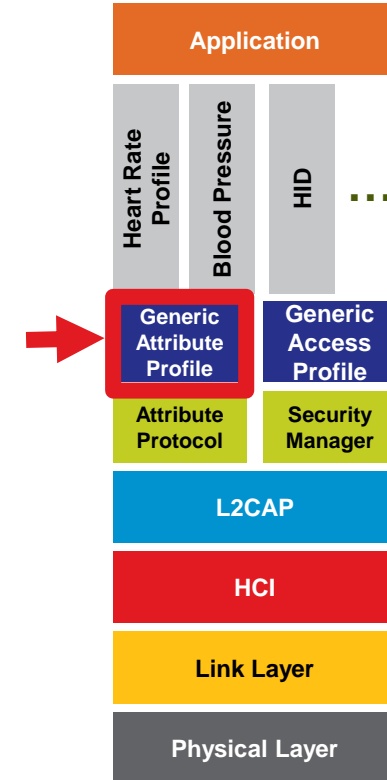
BLE Generic Attribute Profile (GATT)

GATT can be defined as a group of attributes with a common purpose, hence, services were created.

It provides the methods in which the services can be discovered and can be used.

GATT establish a strict hierarchy to organize attributes in a reusable and practical manner.

Allows the access and retrieval of information between client and server to follow a concise set of rules that together constitute the framework used by all GATT-based profiles

BLE Generic Attribute Profile (GATT)

Server

Corresponds to the ATT server

Contains all Attributes

Sends server-initiated updates using indications and notifications

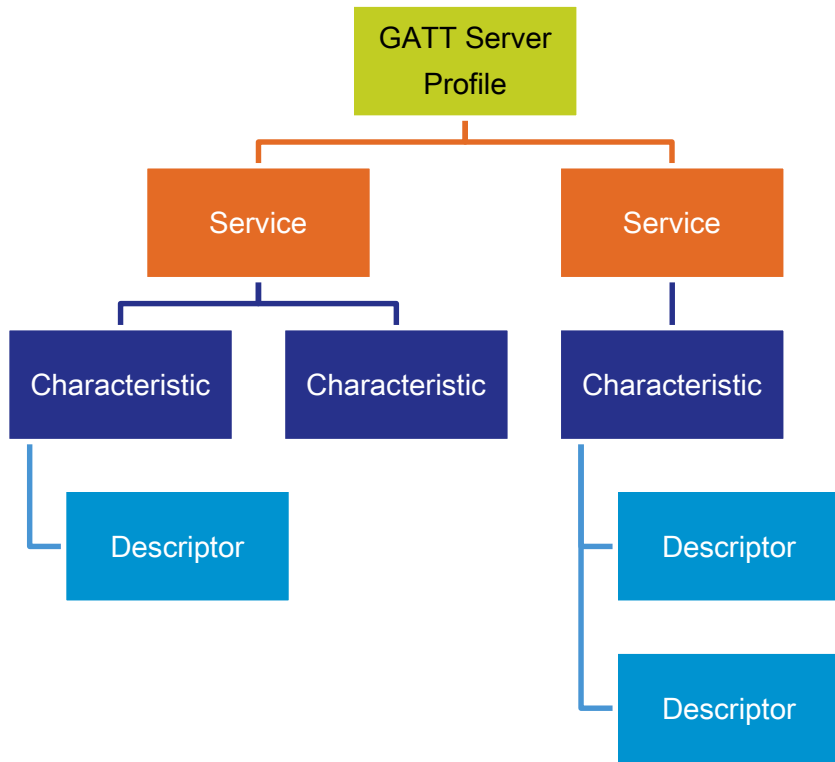Responsible of storing and making the user data available to the client

Client

Corresponds to the ATT client

Sends requests to a server and receives responses

GATT does not know anything in advance about the server's attributes, to find out what they are the Client request a service discovery.

BLE GATT Profile Hierarchy

Profile defines the main behavior of the device. It is formed by a collection of services.

Service is a collection of data and behaviors to accomplish a particular function or feature

Characteristics is a value used in a service. Contains user data.

Descriptor describe the value or allows configuration of the server with respect to the characteristic

BLE GATT Adopted Profiles

Continually updating profile and service support as available from Bluetooth SIG

SpecificationName	SpecificationType	SpecificationLevel
Alert Notification	org.bluetooth.profile.alert_notification	Adopted
Blood Pressure	org.bluetooth.profile.blood_pressure	Adopted
Cycling Power	org.bluetooth.profile.cycling_power	Adopted
Cycling Speed and Cadence	org.bluetooth.profile.cycling_speed_and_cadence	Adopted
Find Me	org.bluetooth.profile.find_me	Adopted
Glucose	org.bluetooth.profile.glucose	Adopted
Health Thermometer	org.bluetooth.profile.health_thermometer	Adopted
Heart Rate	org.bluetooth.profile.heart_rate	Adopted
HID OVER GATT	org.bluetooth.profile.hid_over_gatt	Adopted
Location and Navigation	org.bluetooth.profile.location_and_navigation	Adopted
Phone Alert Status	org.bluetooth.profile.phone_alert_status	Adopted
Proximity	org.bluetooth.profile.proximity	Adopted
Running Speed and Cadence	org.bluetooth.profile.running_speed_and_cadence	Adopted
Scan Parameters	org.bluetooth.profile.scan_parameters	Adopted
Time	org.bluetooth.profile.time	Adopted

https://developer.bluetooth.org/gatt/profiles/Pages/ProfilesHome.aspx

Heart Rate Sensor – gatt_db.h

- Heart Rate Primary Service

```
PRIMARY_SERVICE(service_heart_rate, gBleSig_HeartRateService_d)

    CHARACTERISTIC(char_hr_measurement, gBleSig_HrMeasurement_d, (gGattCharPropNotify_c))
        VALUE_VARLEN(value_hr_measurement, gBleSig_HrMeasurement_d, (gPermissionNone_c), 22,
            2, 0x00, 0xB4)
        CCCD(cccd_hr_measurement)

    CHARACTERISTIC(char_body_sensor_location, gBleSig_BodySensorLocation_d,
                    (gGattCharPropRead_c) )
        VALUE(value_body_sensor_location, gBleSig_BodySensorLocation_d,
            (gPermissionFlagReadable_c), 1, 0x01)

    CHARACTERISTIC(char_hr_ctrl_point, gBleSig_HrControlPoint_d, (gGattCharPropWrite_c) )
        VALUE(value_hr_ctrl_point, gBleSig_HrControlPoint_d,
                (gPermissionFlagWritable_c), 1, 0x00)
```

Heart Rate Sensor – gatt_db.h (2)

- Device Information Primary Service

```
PRIMARY_SERVICE(service_device_info, gBleSig_DeviceInformationService_d)

    CHARACTERISTIC(char_manuf_name, gBleSig_ManufacturerNameString_d, (gGattCharPropRead_c) )
        VALUE(value_manuf_name, gBleSig_ManufacturerNameString_d, (gPermissionFlagReadable_c), 9, "Freescale")

    CHARACTERISTIC(char_model_no, gBleSig_ModelNumberString_d, (gGattCharPropRead_c) )
        VALUE(value_model_no, gBleSig_ModelNumberString_d, (gPermissionFlagReadable_c), 8, "HRS Demo")

    CHARACTERISTIC(char_serial_no, gBleSig_SerialNumberString_d, (gGattCharPropRead_c) )
        VALUE(value_serial_no, gBleSig_SerialNumberString_d, (gPermissionFlagReadable_c), 7, "BLESN01")

    CHARACTERISTIC(char_hw_rev, gBleSig_HardwareRevisionString_d, (gGattCharPropRead_c) )
        VALUE(value_hw_rev, gBleSig_HardwareRevisionString_d, (gPermissionFlagReadable_c), sizeof(BOARD_NAME),
            BOARD_NAME)

    CHARACTERISTIC(char_fw_rev, gBleSig_FirmwareRevisionString_d, (gGattCharPropRead_c) )
        VALUE(value_fw_rev, gBleSig_FirmwareRevisionString_d, (gPermissionFlagReadable_c), 5, "1.1.1")

    CHARACTERISTIC(char_sw_rev, gBleSig_SoftwareRevisionString_d, (gGattCharPropRead_c) )
        VALUE(value_sw_rev, gBleSig_SoftwareRevisionString_d, (gPermissionFlagReadable_c), 5, "1.1.3")

    CHARACTERISTIC(char_system_id, gBleSig_SystemId_d, (gGattCharPropRead_c) )
        VALUE(value_system_id, gBleSig_SystemId_d, (gPermissionFlagReadable_c), 8, 0x00, 0x00, 0x00, 0xFE,
            0xFF, 0x9F, 0x04, 0x00)

    CHARACTERISTIC(char_rcdl, gBleSig_IeeeRcdl_d, (gGattCharPropRead_c) )
        VALUE(value_rcdl, gBleSig_IeeeRcdl_d, (gPermissionFlagReadable_c), 4, 0x00, 0x00, 0x00, 0x00)
```

Heart Rate Sensor – gatt_db.h (2)

- GATT Primary Service

```
PRIMARY_SERVICE(service_gatt, gBleSig_GenericAttributeProfile_d)
    CHARACTERISTIC(char_service_changed, gBleSig_GattServiceChanged_d, (gGattCharPropRead_c |
                        gGattCharPropNotify_c) )
        VALUE(value_service_changed, gBleSig_GattServiceChanged_d, (gPermissionNone_c), 4, 0x00, 0x00,
                        0x00, 0x00)
        CCCD(cccd_service_changed)
```

GAP Primary Service

```
PRIMARY_SERVICE(service_gap, gBleSig_GenericAccessProfile_d)

    CHARACTERISTIC(char_device_name, gBleSig_GapDeviceName_d, (gGattCharPropRead_c) )
        VALUE(value_device_name, gBleSig_GapDeviceName_d, (gPermissionFlagReadable_c), 11, "FSL_BLE_HRS")

    CHARACTERISTIC(char_appearance, gBleSig_GapAppearance_d, (gGattCharPropRead_c) )
        VALUE(value_appearance, gBleSig_GapAppearance_d, (gPermissionFlagReadable_c), 2,
          UuidArray(gGenericHeartrateSensor_c))

    CHARACTERISTIC(char_ppcp, gBleSig_GapPpcp_d, (gGattCharPropRead_c) )
        VALUE(value_ppcp, gBleSig_GapPpcp_d, (gPermissionFlagReadable_c), 8, 0x0A, 0x00, 0x10, 0x00, 0x64,
          0x00, 0xE2, 0x04)
```

Custom Service

A custom service might be useful when non of the adopted fits your application

Custom services and characteristics must use 128-bit UUID

Considerations for a custom service

1. Define the functions to be performed by the new service

2. Define the characteristics to be used, its value, property and permissions

3. Create descriptors if necessary

4. Create the custom service on the BLE stack

Summary BLE Roles

Depending on the layer, the device roles uses different names:

At Link layer
    Master
    Slave

At GAP (non-beacon)
    Central
    Peripheral

At GATT
    Server
    Client

HRS DEMO

NXP Kinetis Wireless Solutions Agenda

- **KW41 Family Overview**

- **Bluetooth Low Energy (BLE)**

- **Thread**

- **BLE+Thread**

- **Wireless Framework**

- **Modular Gateway**

THREAD

INTRO TO THREAD

The Need For a New Wireless Network

More and more products are being connected in the home

- – Direct internet access

- – Simple to add and remove from the network

- – Must be secure

- – Robust

- – Battery operated for years

Existing protocols did not meet these requirements

Thread Group was formed by 7 companies to solve the probl
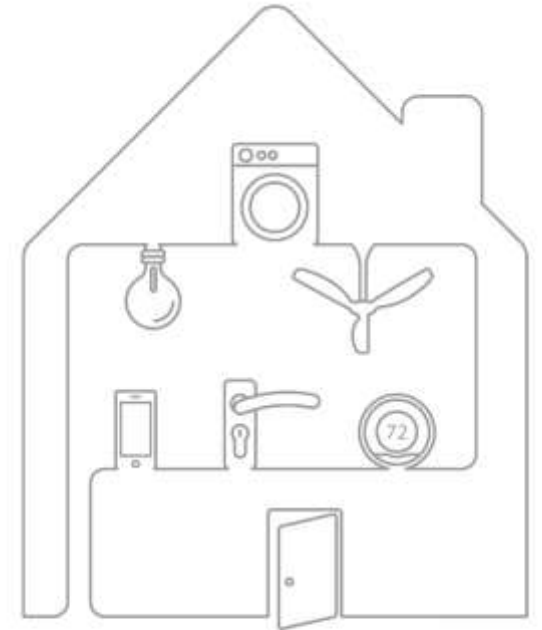
A secure low-power wireless IPv6 mesh networking protocol

- Open, worldwide protocol built on top of 802.15.4
- Every node has a unique IPv6 address
- Low power end nodes
- Simple for consumers to add or remove nodes
- Scalable up to 250+ nodes
- Secure – AES128 encryption
- No single point of failure
- Network operates without cloud connection
- Fast time to market
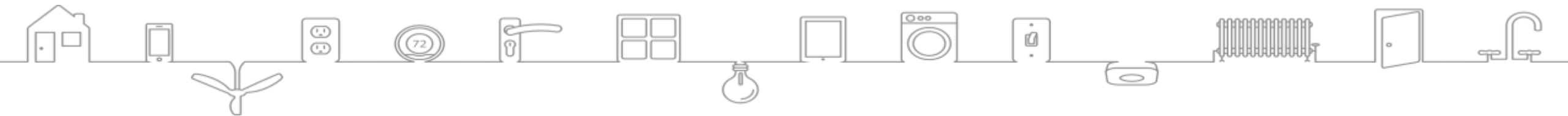- Only IP-based mesh networking protocol available
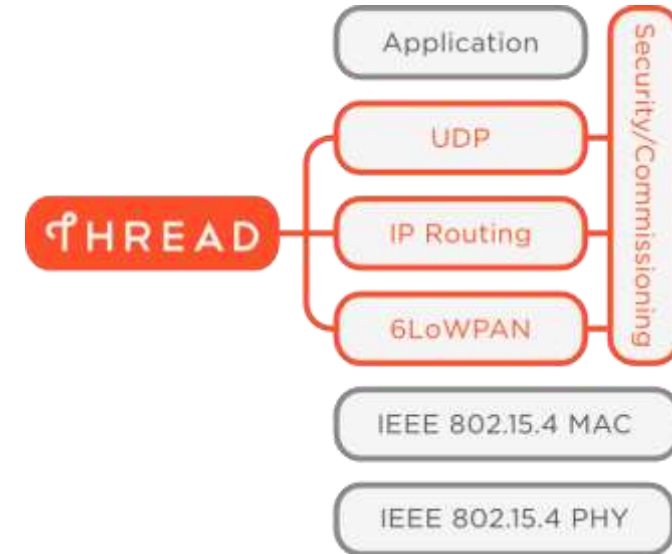
Target Applications

Thread is designed for all sorts of products in the home

- Appliances
- Access control
- Climate control
- Energy management
- Lighting
- Safety
- Security
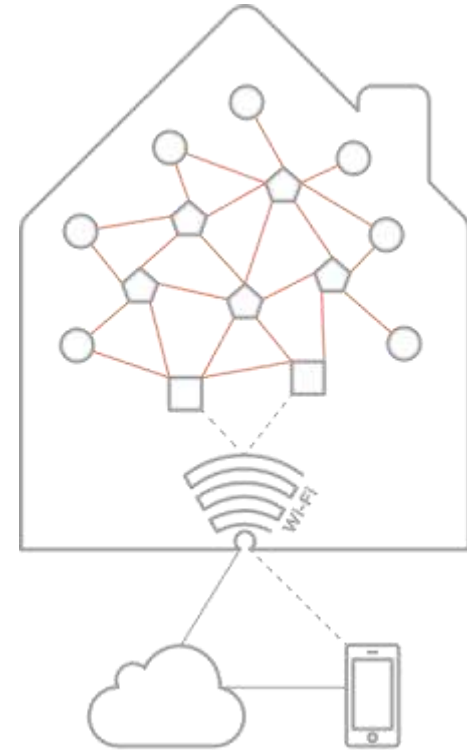
Devices working together to form a cohesive mesh network

About Thread Group

7 Founding Companies, grown to 12 Sponsor Companies, 230+ member 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 Ericsson - 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: Jean-Michel Orsat - Somfy

Director: Greg Blackett – Tyco

Thread Certification

All Thread devices will require network certification to use Thread certified logo on commercial products
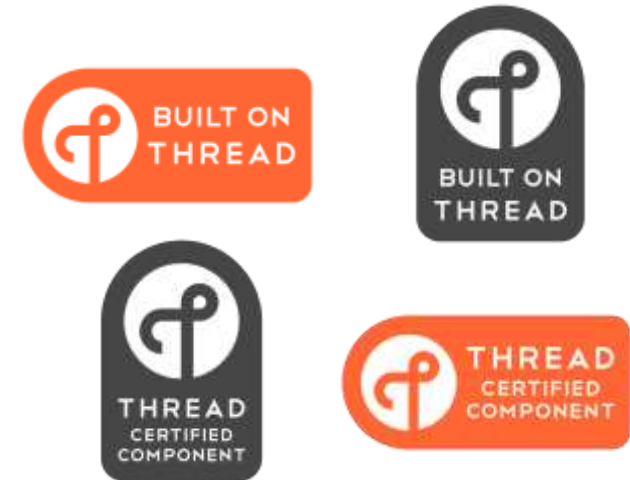
Validation of device behavior
- Commissioning
- Network functionality and interoperability
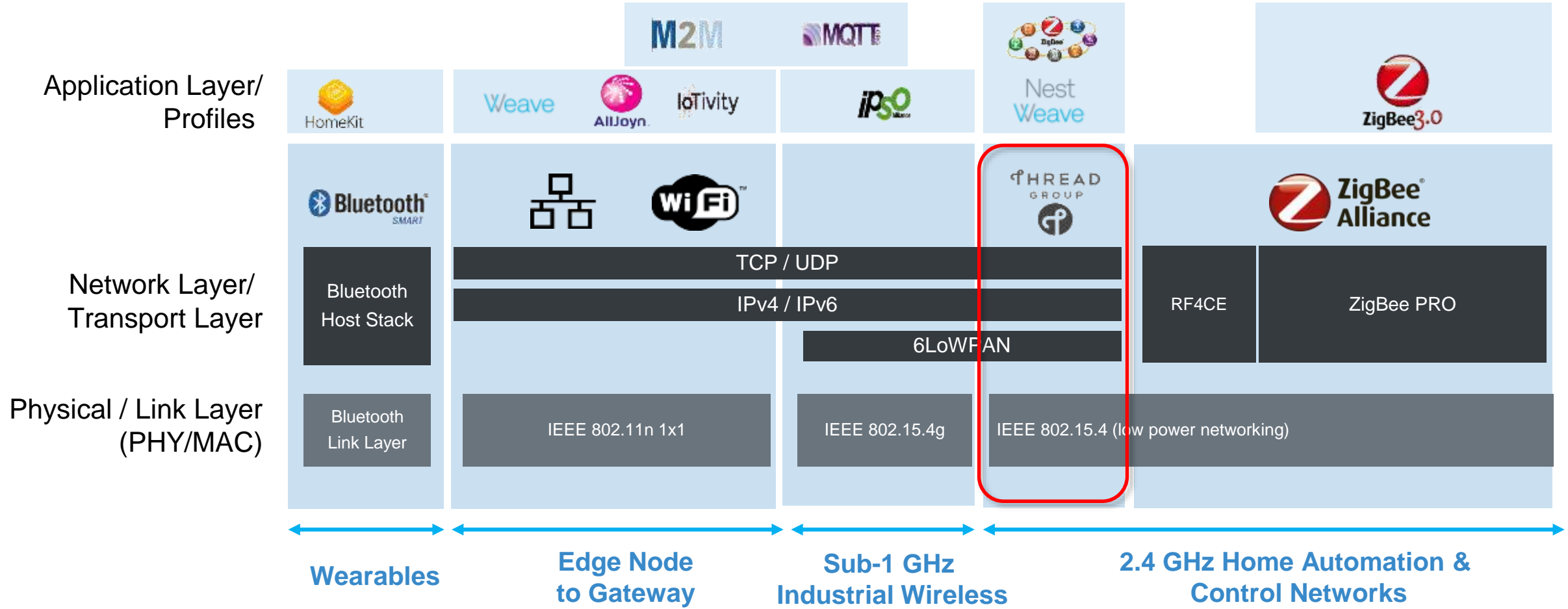- Device operation in network

The certification program addresses both components and end products

Sponsor and Contributor Members have access to standard test harness and sample commissioning app

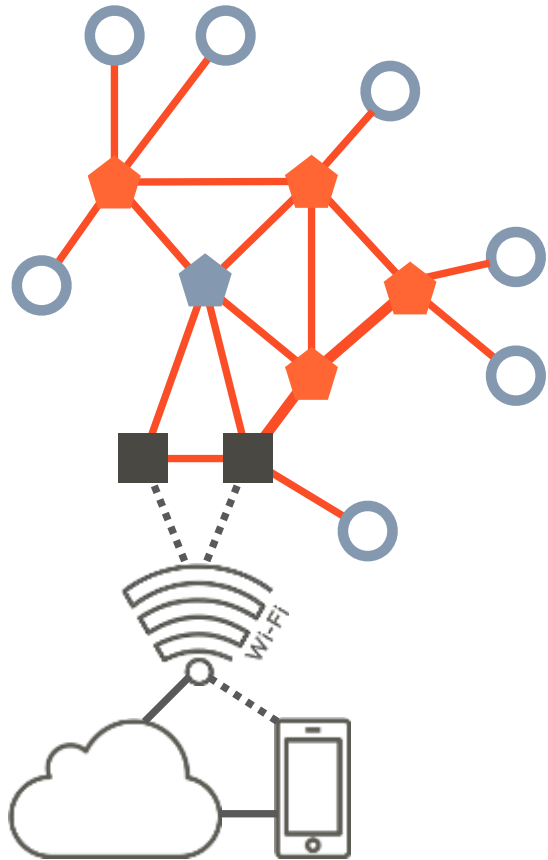Certification through an approved 3rd party test lab

IoT Connectivity Landscape – Where does Thread play?

THREAD NETWORKING ARCHITECTURE

NXP

Network Architecture

○ End Device or Router Eligible End Device

⬠ Active Router

⬠ Leader

◼ Border Router

— Thread Network Link

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
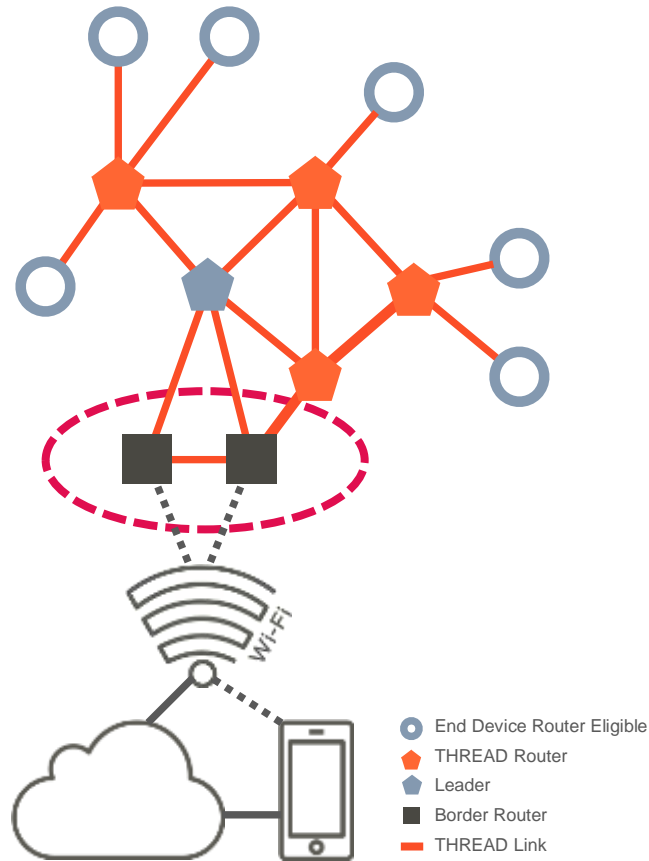Thread Routers form backbone of the Thread mesh
Leader-eligible

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)

Thread Border Router

End Device Router Eligible
THREAD Router
Leader
Border Router
THREAD Link

## The Border Router

- Provides a bridge between a Thread network and a home LAN or other upstream IP infrastructure.

- Is usually a superset of Router Eligible Device

- Has at least one additional interface other than IEEE 802.15.4 (e.g.: Wi-Fi, Ethernet, USB)

- Multiple Border Routers can exist in a Thread Network
  - However a border router is not required either.

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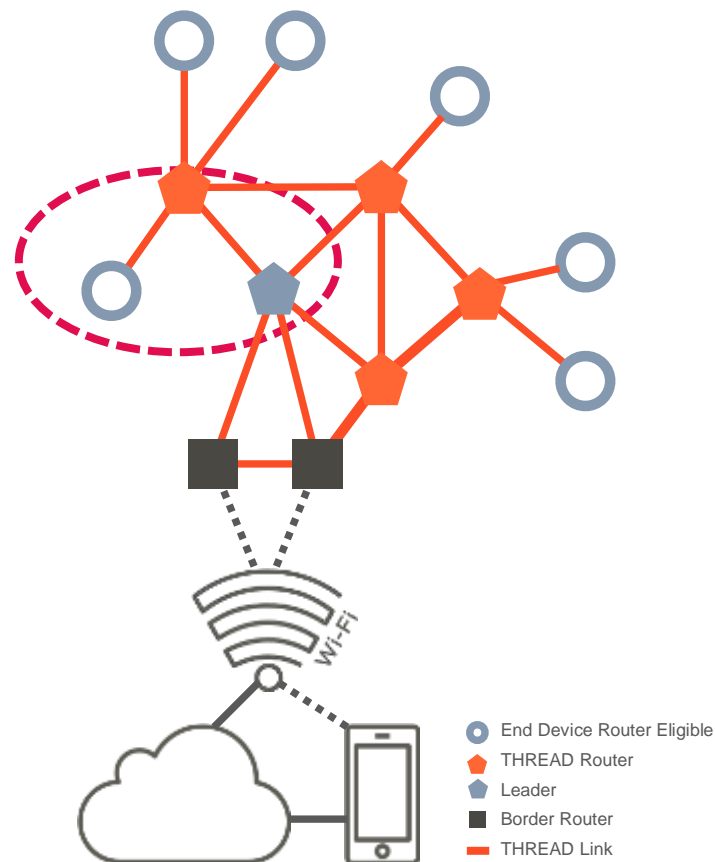
A **Router Eligible End Device** can play multiple roles at runtime

Leader
If it is the initial device in the network partition, or the node selected when the original leader becomes 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
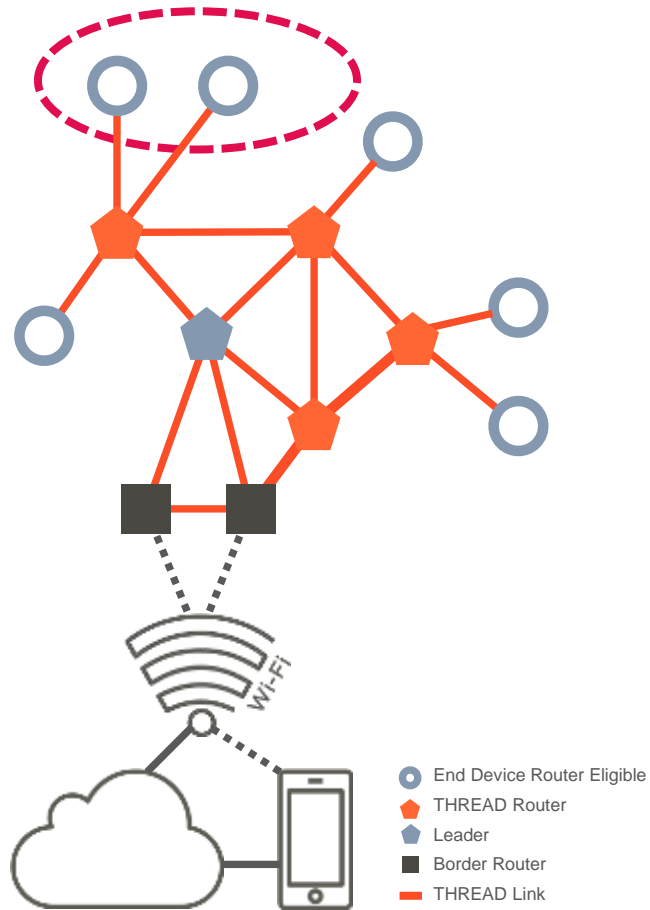
Active Router
A REED requests the Leader for it to become an Active Router when the network has relatively limited connectivity. e.g.: when total number of existing Active Routers is < 16

Legend:
- End Device Router Eligible
- THREAD Router
- Leader
- Border Router
- THREAD Link

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

NXP

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 may **periodically be turned off** or has a high capacity battery with recharge

A **Sleepy End Device**

- Does not have routing capabilities
- "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)

Legend:
- End Device Router Eligible
- THREAD Router
- Leader
- Border Router
- THREAD Link

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

No Single Point of Failure

- No need to recognize specialized devices within the network

- Leader makes decisions but upon failure another Router will become Leader

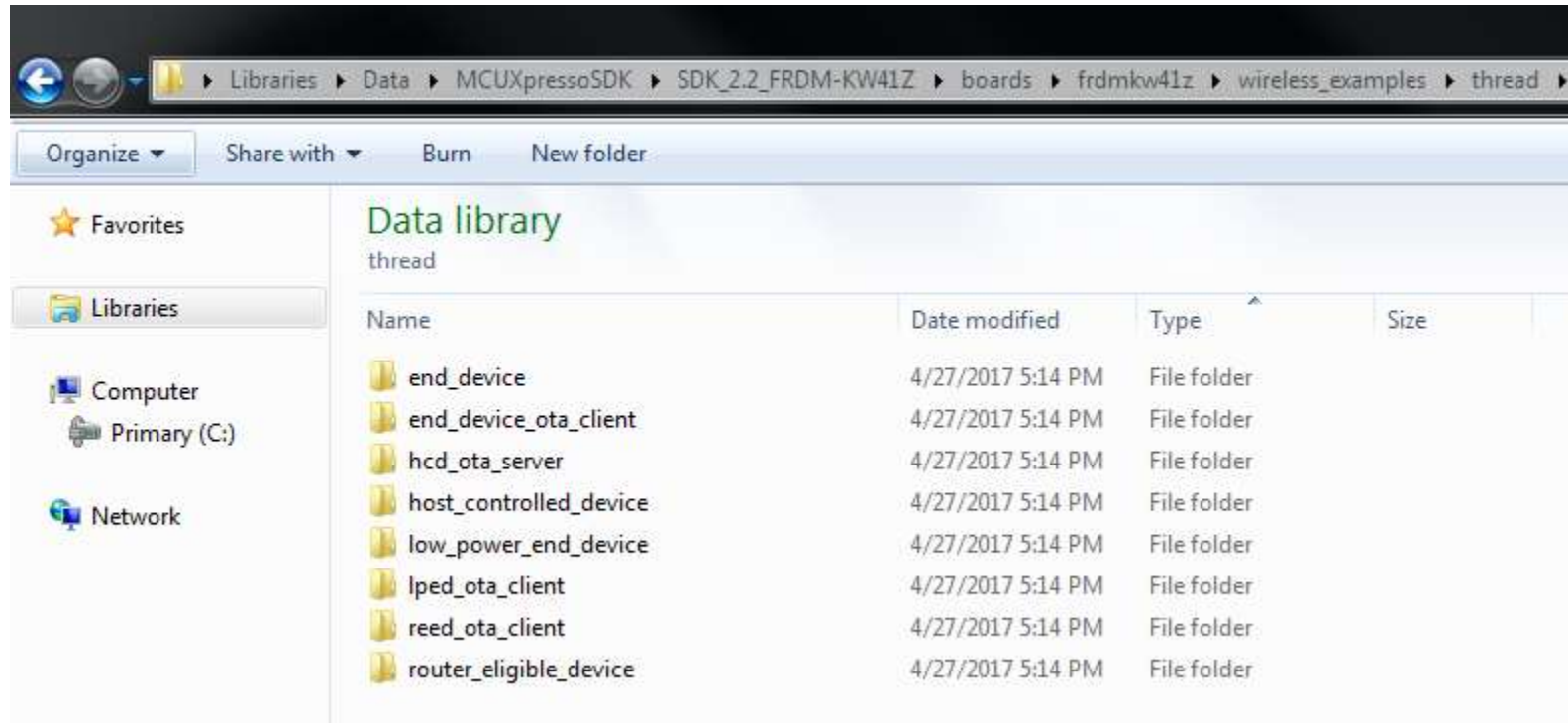- Network will add Routers to improve connectivity when required

End Device Router Eligible
THREAD Router
Leader
Border Router
THREAD Link
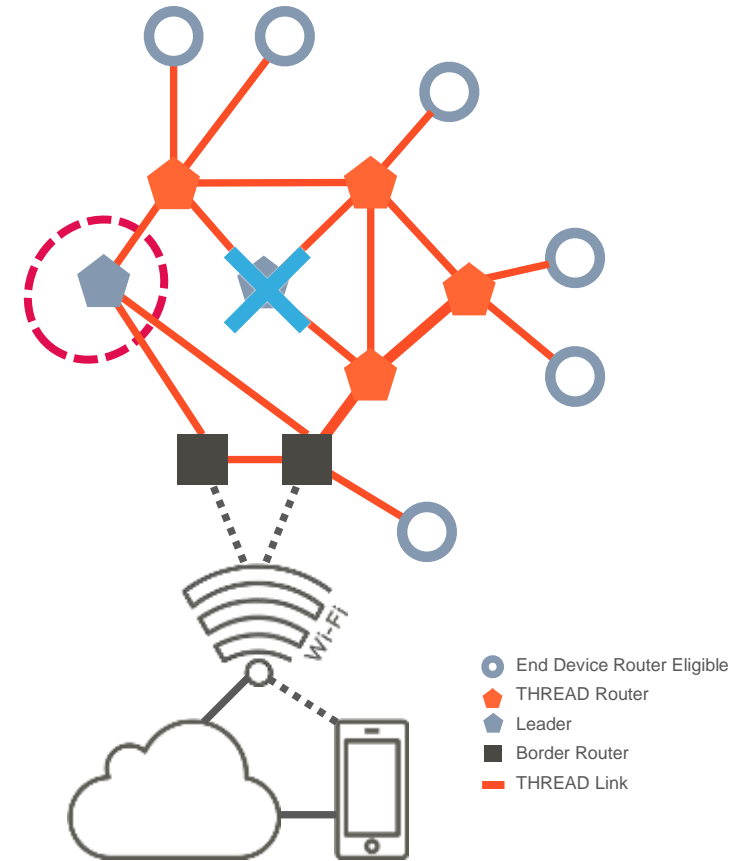
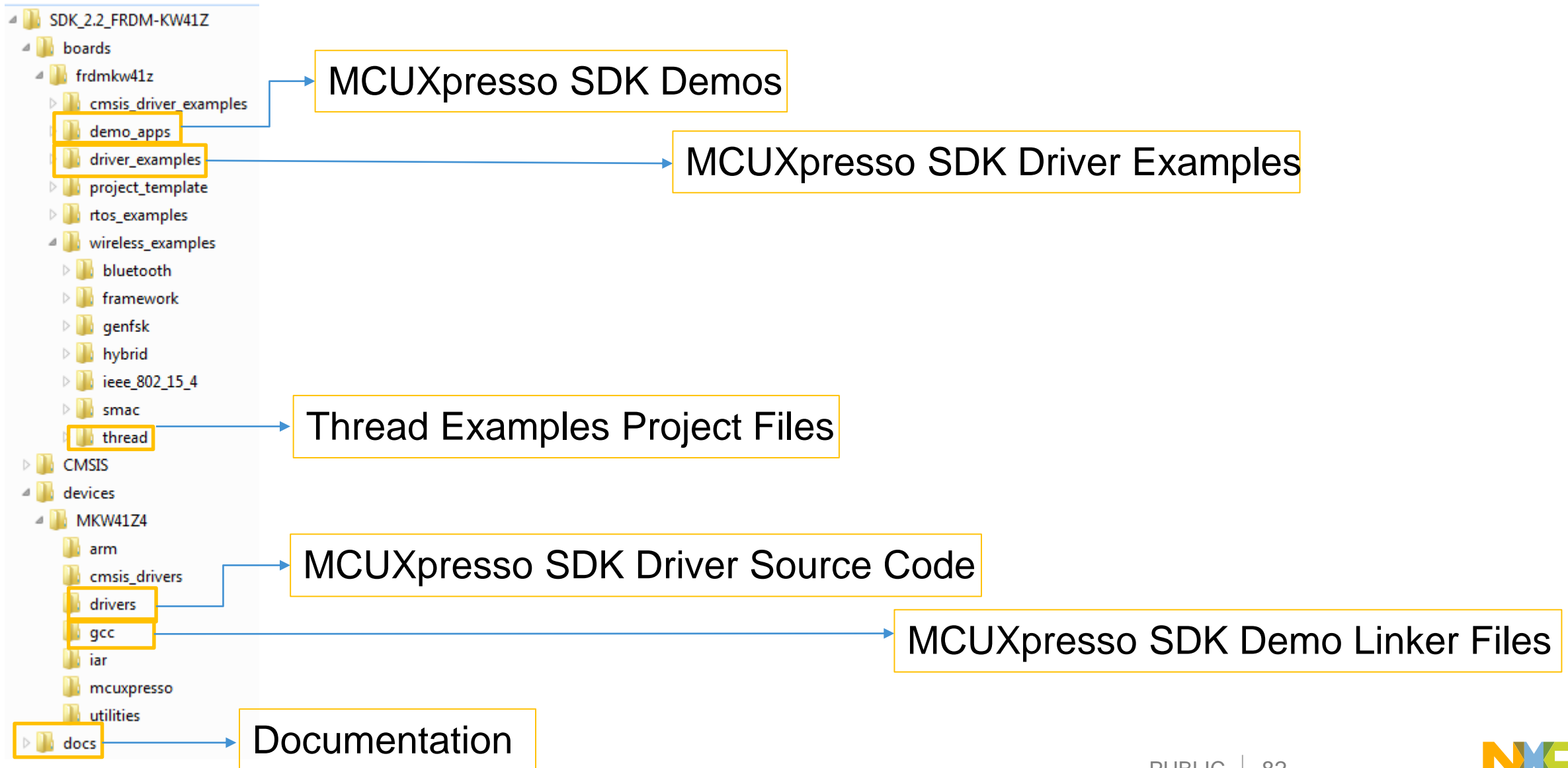# GETTING THREAD

MCUXpresso SDK for KW41Z

- Contains all the KW41Z Wireless Connectivity solution stacks:
  - Thread v1.1
  - IEEE 802.15.4
  - SMAC
  - Bluetooth Low Energy v4.2
  - Generic FSK Link Layer software
- Also contains MCUXpresso SDK sources
- Supports the FRDM-KW41Z and USB-KW41Z development boards
- Supported IDEs:
  - MCUXpresso IDE
  - IAR for ARM
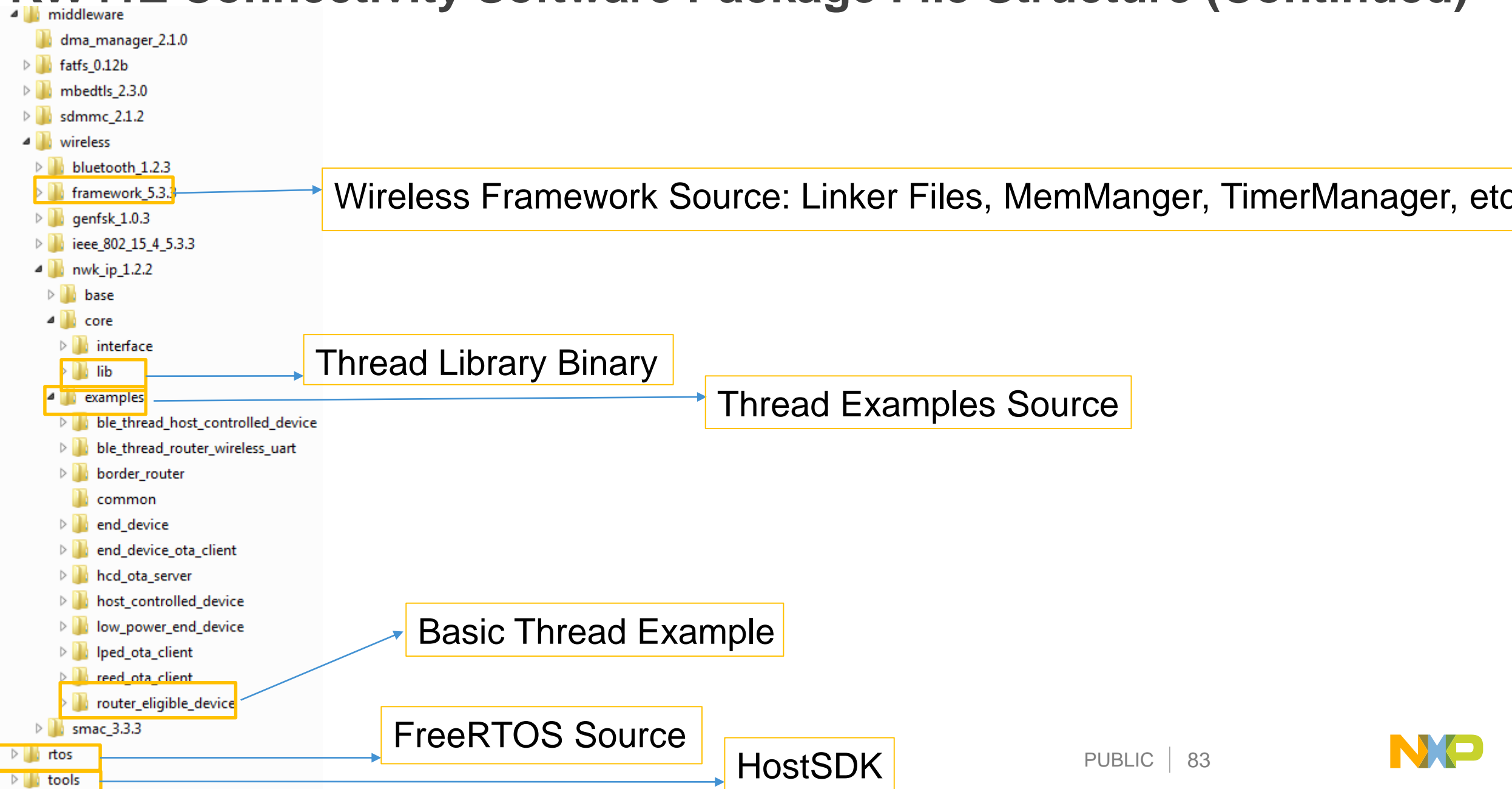- Available Now!

Installation

- Thread examples found at: **\boards\frdmkw41z\wireless_examples\thread**

KW41Z Connectivity Software Package File Structure

MCUXpresso SDK Demos

MCUXpresso SDK Driver Examples

Thread Examples Project Files

MCUXpresso SDK Driver Source Code

MCUXpresso SDK Demo Linker Files

Documentation

KW41Z Connectivity Software Package File Structure (Continued)

- middleware
  - dma_manager_2.1.0
  - fatfs_0.12b
  - mbedtls_2.3.0
  - sdmmc_2.1.2
  - wireless
    - bluetooth_1.2.3
    - framework_5.3.3 → **Wireless Framework Source: Linker Files, MemManger, TimerManager, etc**
    - genfsk_1.0.3
    - ieee_802_15_4_5.3.3
    - nwk_ip_1.2.2
      - base
      - core
        - interface
        - lib → **Thread Library Binary**
        - examples → **Thread Examples Source**
          - ble_thread_host_controlled_device
          - ble_thread_router_wireless_uart
          - border_router
          - common
          - end_device
          - end_device_ota_client
          - hcd_ota_server
          - host_controlled_device
          - low_power_end_device
          - lped_ota_client
          - reed_ota_client
          - router_eligible_device → **Basic Thread Example**
    - smac_3.3.3
  - rtos → **FreeRTOS Source**
  - tools → **HostSDK**

Thread Documentation

- Kinetis Thread Stack Demo Applications User Guide

- Kinetis Thread Stack 1.1 Release Notes

- Kinetis Thread Stack API Reference Manual

- Kinetis Thread Stack and FSCI Bootloader Quick Start Guide

- Kinetis Thread Stack Application Development Guide

- Kinetis Thread Stack OTA Firmware Update User's Guide

- Kinetis Thread Stack Shell Interface User's Guide

- Kinetis Thread Host Control Interface Reference Manual

THREAD DETAILS

So how many addresses does a Thread device get?

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

LL64 interface ID is based on IEEE **802.15.4 Random Extended Address**

At least 3 Unicast IPv6 addresses to the Thread Interface:

Link local address (LL64): **fe80**::**b86f:ab39:9875:7a33**

Mesh local address (ML16, RLOC): **fd9b:91a1:52d5:e428**::ff:fe00:**400**

Mesh local address (ML64, ML-EID): **fd9b:91a1:52d5:e428**::**b457:d8d4:dd35:aea0**

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

Mesh Local Endpoint Identifier ML-EID interface ID is random

Mesh Local Prefix **MLP** is **consistent for a Thread network**, the value is generated according to IPv6 RFC 4193

Two All Thread Nodes multicast addresses:

Link local all Thread Nodes(Multicast): **ff32** 40:fd**9b:91a1:52d5:e428**::**1**

Realm local all Thread Nodes(Multicast): **ff33** 40:fd**9b:91a1:52d5:e428**::**1**

Scope
2 – Link local
3 – Realm Local

Group ID
1 – Multicast all nodes
2 – Multicast all routers

Optional 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**

Unique Local Prefix is defined in each Border Router

Unique Local Address ULA is random

Global Prefix retrieved from the ISP via prefix delegation

Global Unicast Address GUA assigned via DHCPv6 or random if assigned via SLAAC

Thread internal only!!!

OK to be used by application

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

Thread Scopes

Scopes specify the boundaries of networks when using and forwarding packets for an address

Link Local single-hop within radio range

Mesh Local multi-hop within the PAN

Unique Local multi-hop within the PAN and inter-PAN for the same network

Global internet addressable

http://www.iana.org/assignments/ipv6-address-space/ipv6-address-space.xhtml
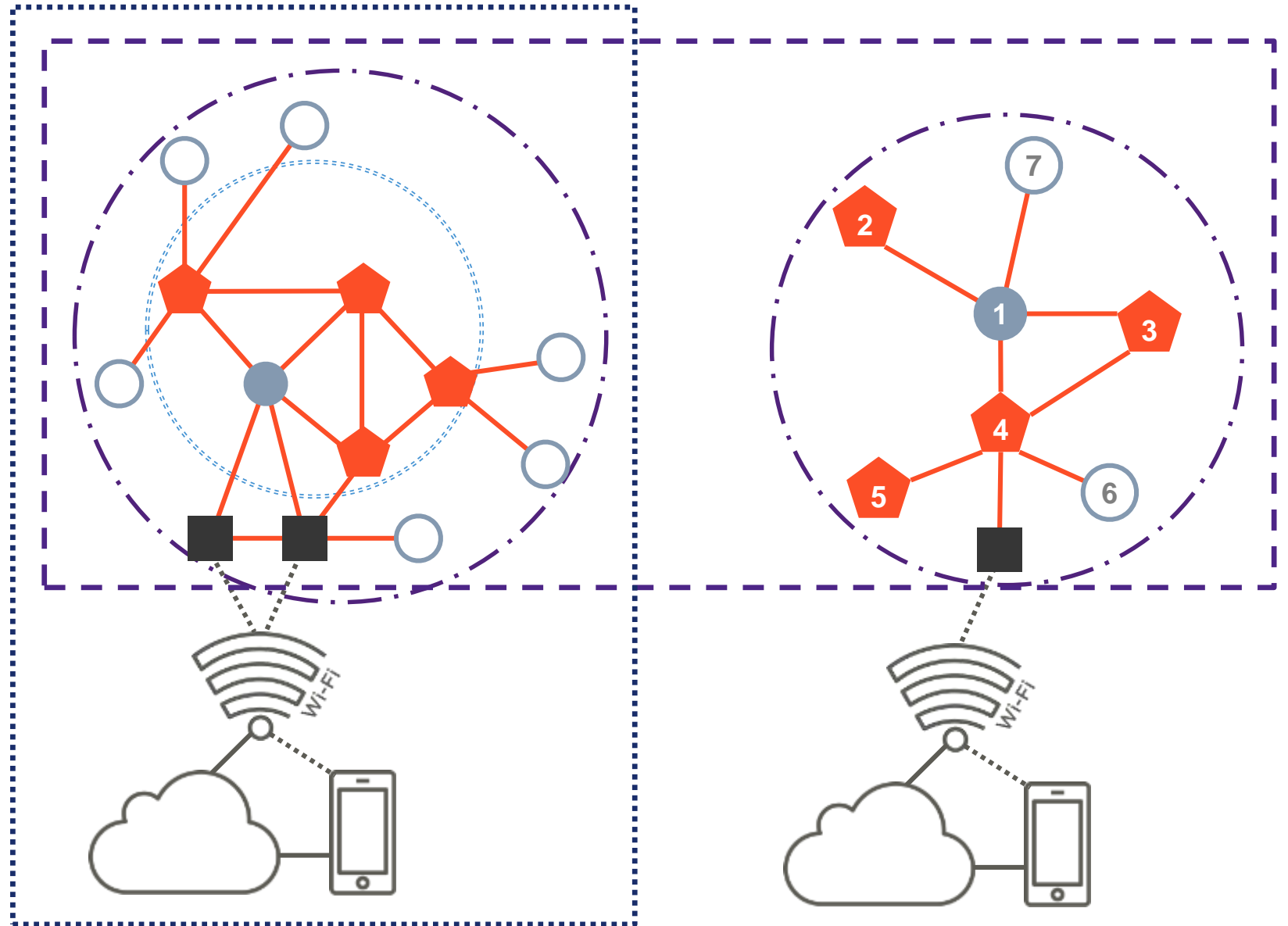
Multicast

- Realm local all Thread nodes

Packets can reach every node of the network as long
maximum of 2 "hops" away from the requester

The packet gets forwarded three more times every tim
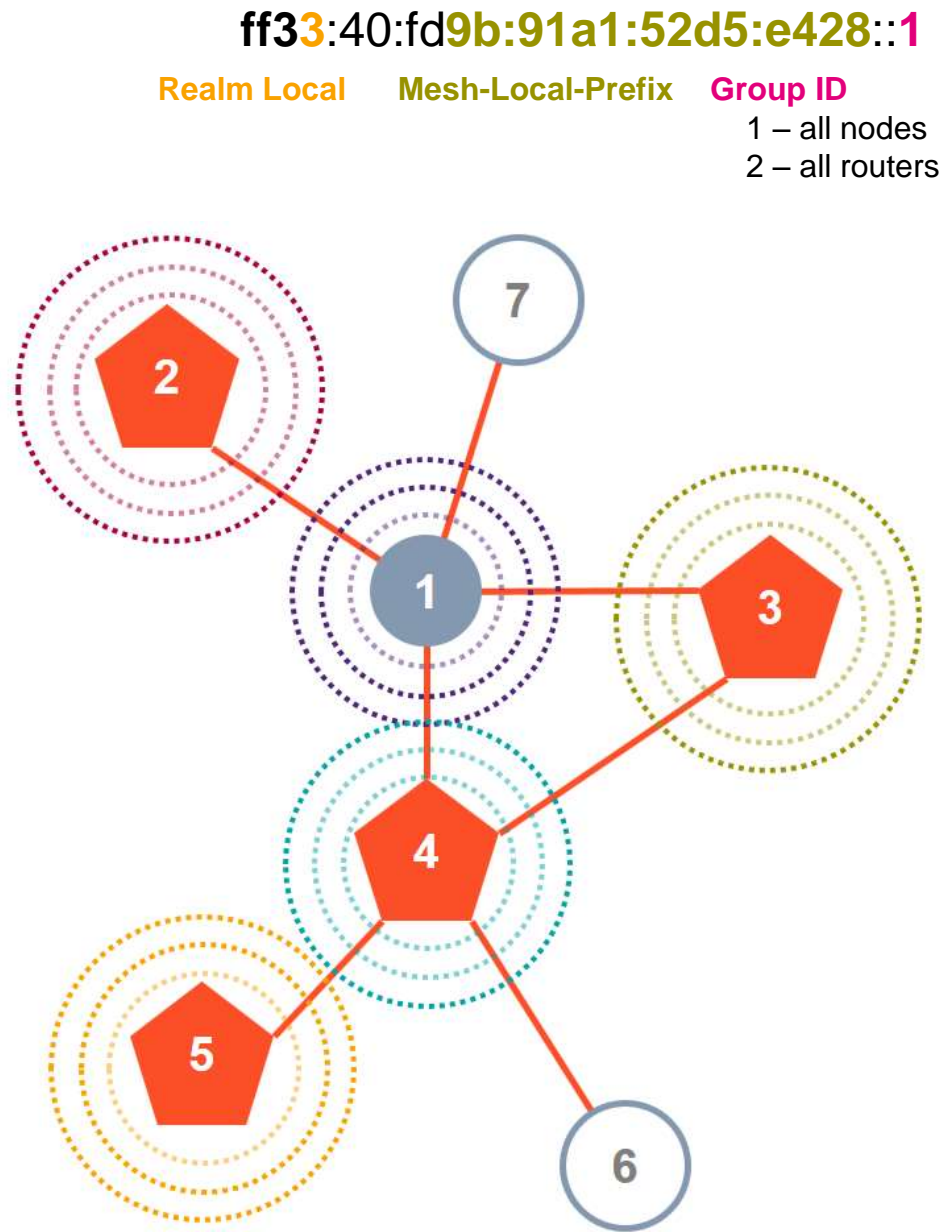
THREAD LAYERS

Thread End Device - High Level Block Diagram
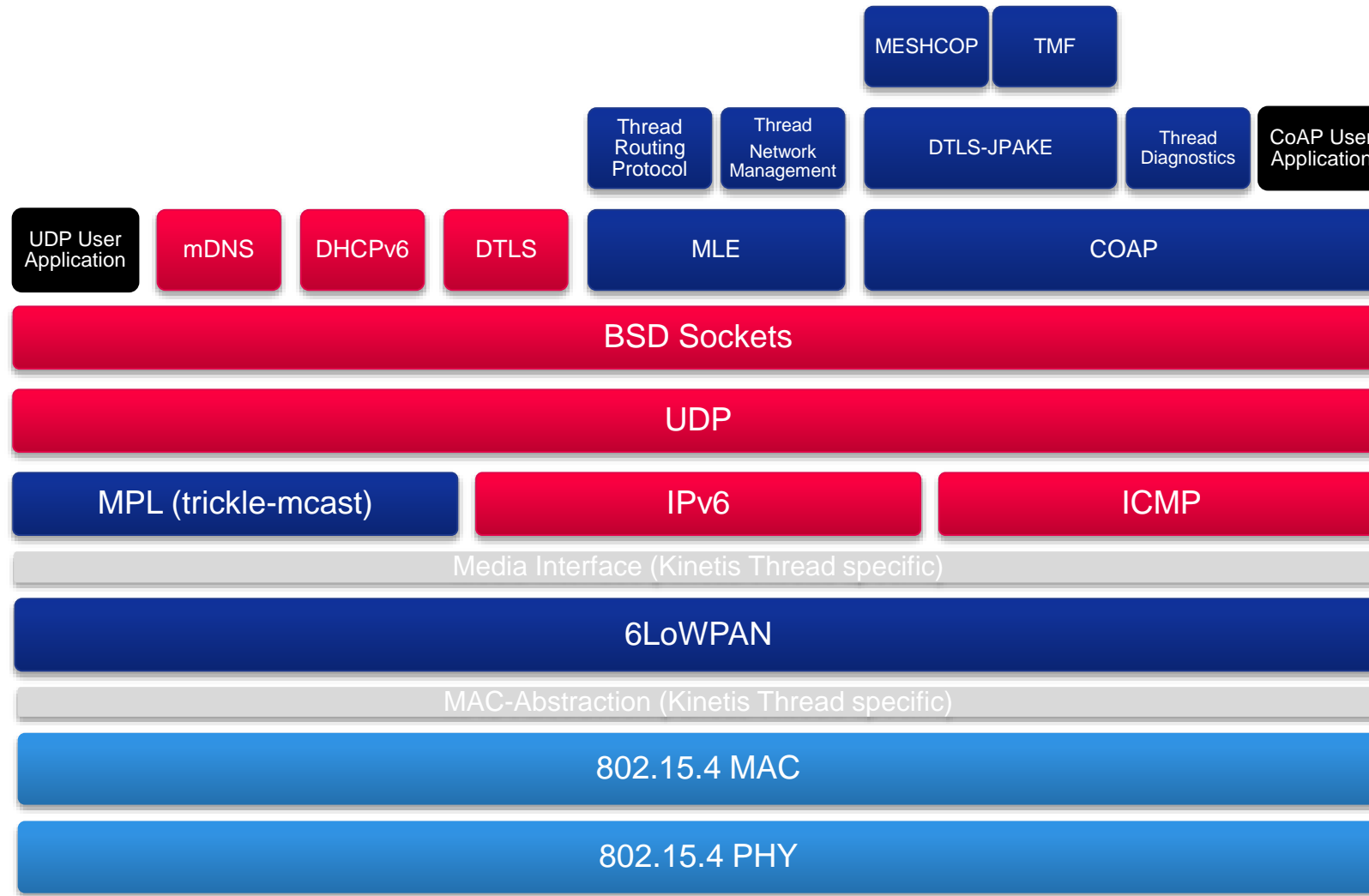
802.15.4 AND 6LOWPAN

IEEE 802.15.4 - PHY

IEEE 802.15.4 channel occupancy on 2.4GHz
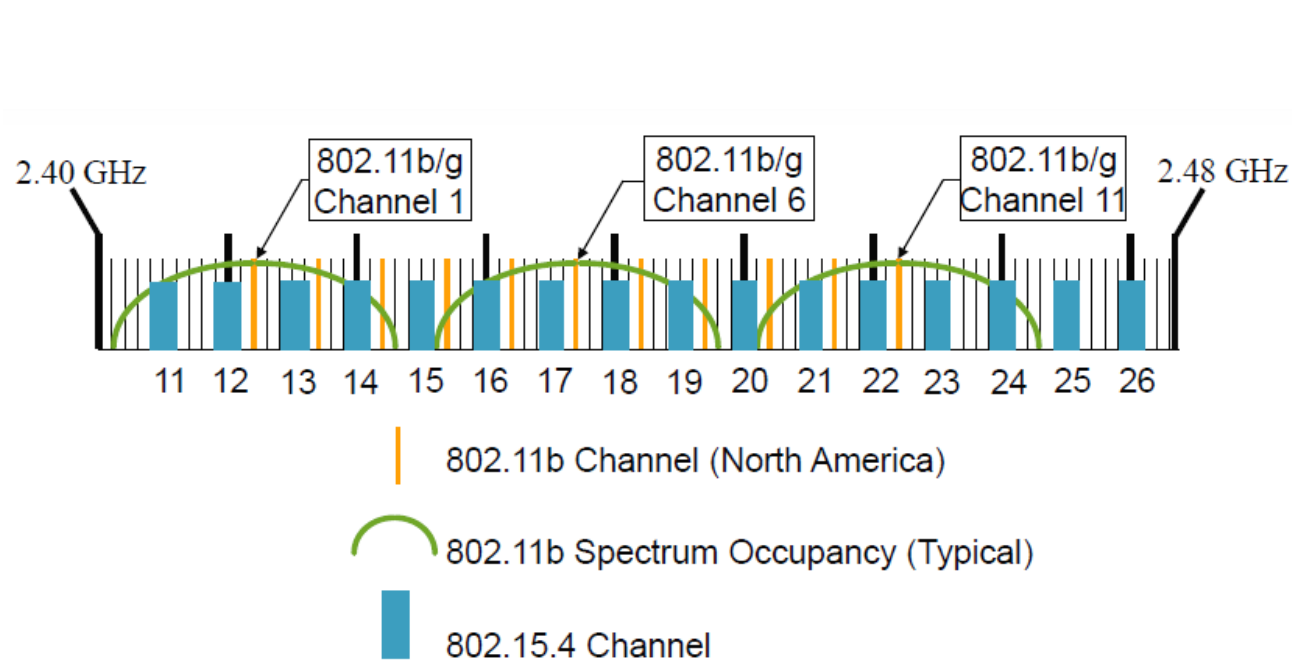
802.15.4 open channels when Wi-Fi fully utilized the band
– 15, 20, 25, 26.

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

NXP

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

2 bytes	1 byte	0-20 bytes	0-14 bytes	Variable	2 bytes
Frame Control	Sequence Number	Addressing Fields	Auxiliary Security Header	Data Payload	FCS

| | | MAC Header | | MAC Payload | MAC Footer |

| PHY Header | | | PHY Payload | | |

127 Bytes

Multi-Hop, Fragmented Packet, 1st Fragment

| 802.15.4 PHY/MAC hdr | Mesh hdr | Fragmentation hdr | Compression hdr | 6LoWPAN Payload |

Multi-Hop, Fragmented Packet, Next Fragments

| 802.15.4 PHY/MAC hdr | Mesh hdr | Fragmentation hdr | 6LoWPAN Payload |

THREAD MANAGEMENT LAYER

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

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

COMMISIONING

Commissioning

MeshCoP or Mesh Commissioning is a protocol for securely authentication, commissioning and joining new untrusted radio devices to a mesh network

Adding new devices it's the process of a human administrator
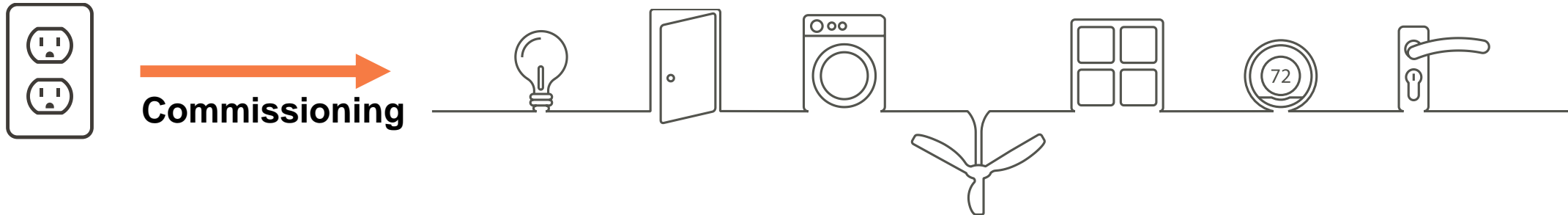
Commissioning

Commissioning Roles

Commissioner currently elected authentication server and the authorizer to provide network credentials **C**

Border Agent any device capable of relaying messages between a Thread Network and a Commissioner **BA**

Commissioner Candidate a device capable of becoming a commissioner and either intends or is currently petitioning the leader to become the Commissioner

CC

Joiner the device to be added by a human administrator to a commissioned Thread network **J**

Joiner Router existing Thread router or REED in the secure Thread Network that is one hop away from the **Joiner**
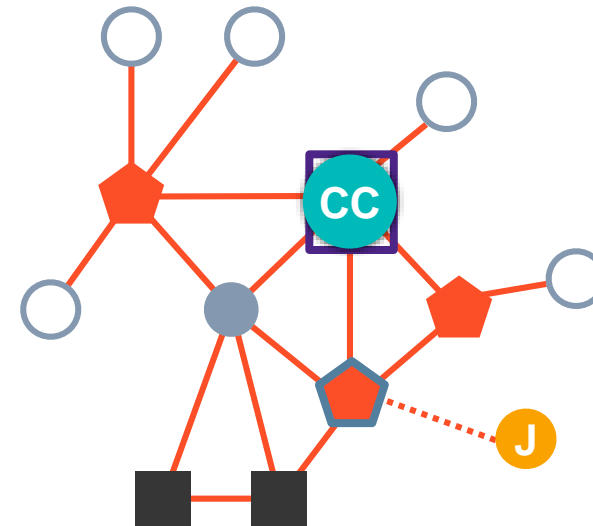
THREAD SECURITY

Security Overview

Commissioning
  Stablish a secure session between a commissioner and a joining device
  DTLS with **elliptic curve J-PAKE** to authenticate and provide network credentials

MAC & Mesh Management Layers
  AES128 encryption for all messages as defined by IEEE 802.15.4 specification

Thread Network
  Automatic **key rotation** mechanisms for parameterizing, changing, and negotiating shared network keys change after a time interval

Generic IP layer
  Capable of carrying multiple **DTLS** or **TLS** flavors

NXP KINETIS & I.MX THREAD PLATFORMS

NXP's Thread Hardware Offering
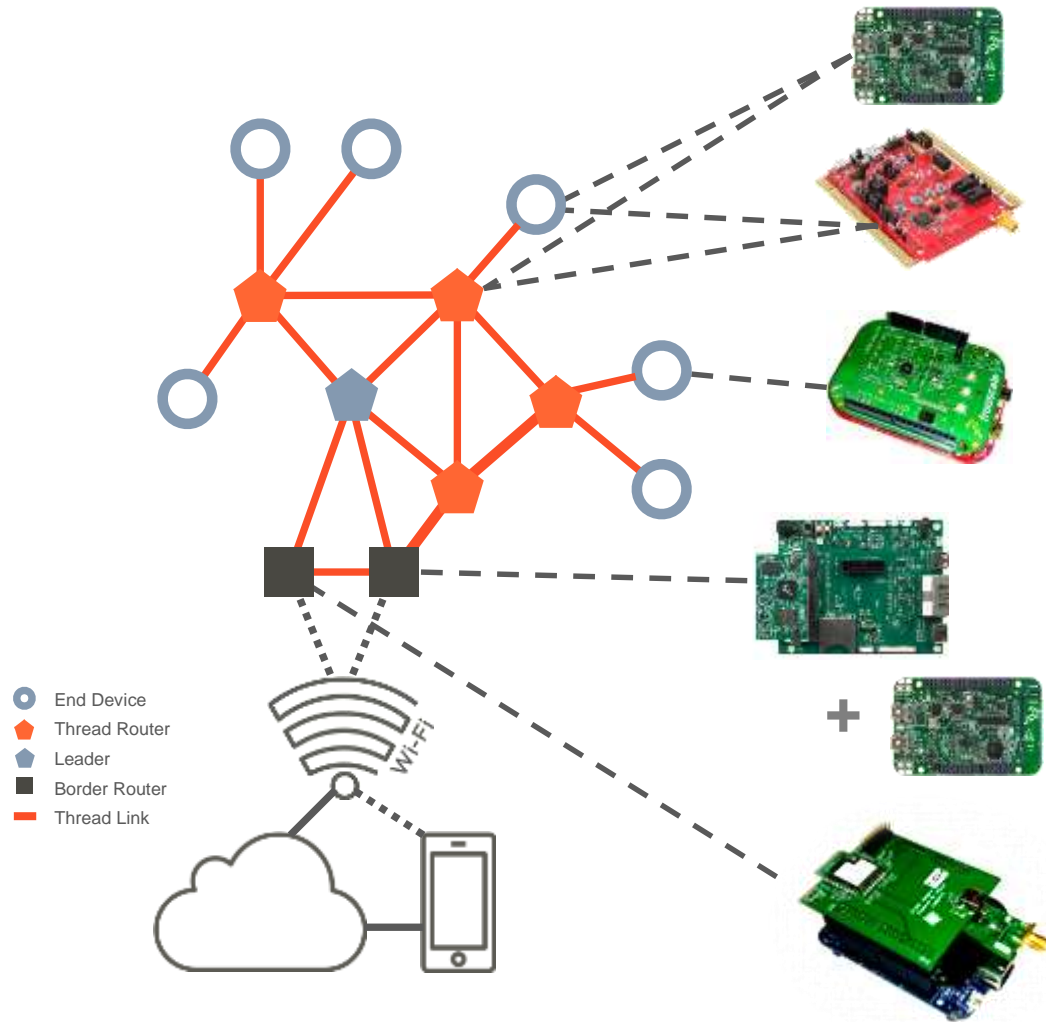
NXP Kinetis KW2xD, KW41Z
Thread Router / REED / End Device
Tower Board and Freedom Board
Kinetis SDK and FreeRTOS

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

NXP i.MX6 UltraLite EVK +
FRDM-KW24D or FRDM-KW41Z
Thread Border Router / Cloud gateway
Provides IP data routing and infrastructure integration
i.MX6UL EVK & Freedom Board
Runs Linux operating system

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

NXP's KW41Z and KW2xD Thread Stacks are **Now Certified!**

Legend:
- End Device
- Thread Router
- Leader
- Border Router
- Thread Link

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

KW41Z/21Z Thread Device Type Code Estimates

End Device
256kB Flash
30kB RAM

Router Device
320kB Flash
50kB RAM

Border Router
320kB Flash
50kB RAM

SPI / UART
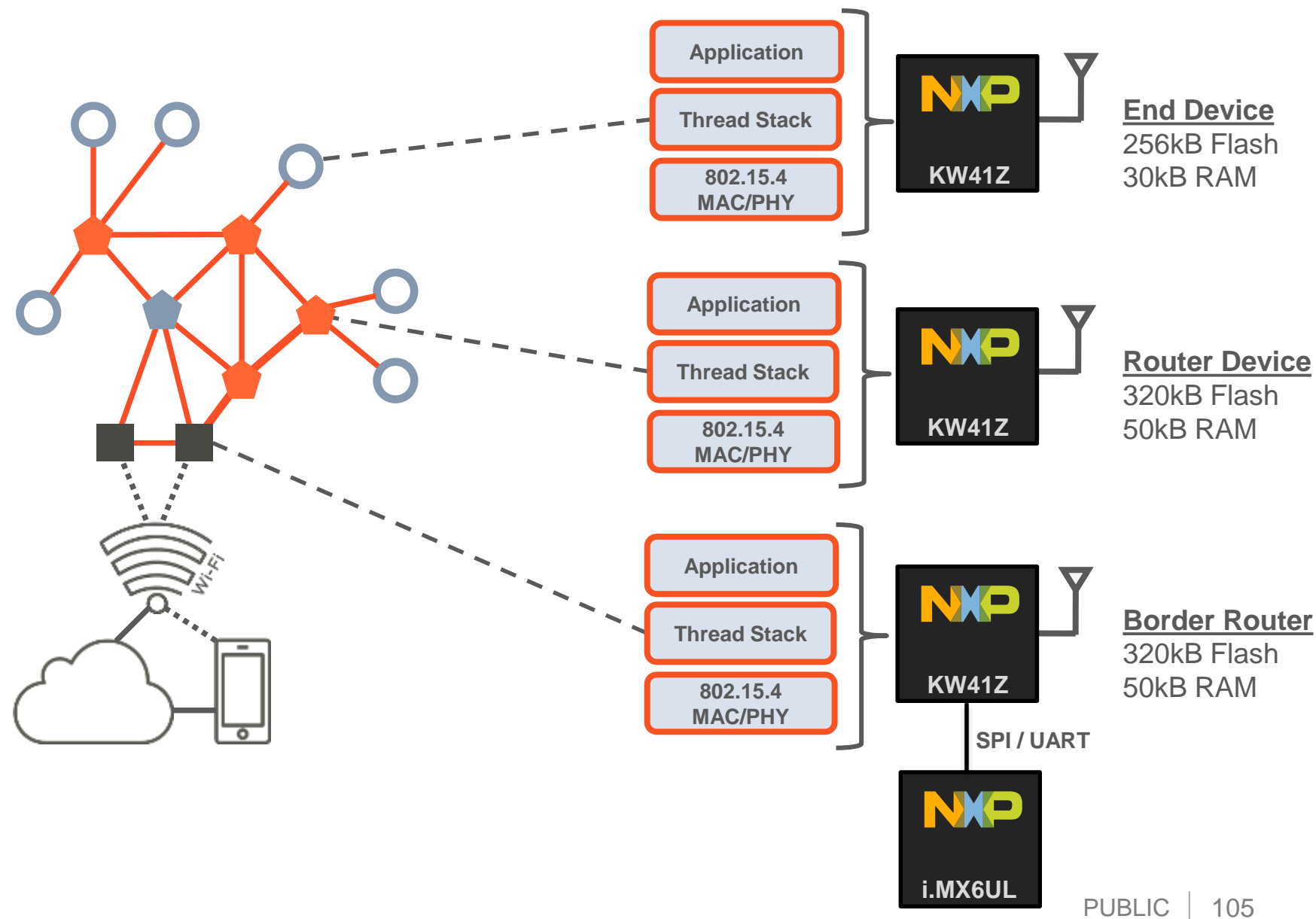
LINKS OF INTEREST

Links of interest

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

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

THREAD DEMO

NXP Kinetis Wireless Solutions Agenda

- **KW41 Family Overview**

- **Bluetooth Low Energy (BLE)**

- **Thread**

- **BLE+Thread**

- **Wireless Framework**

- **Modular Gateway**

BLE+THREAD

Introduction

KW41 deals with three types of coexistence

- Internal: Two protocols in the same chip (i.e. BLE + 802.15.4)

- Inter-IC: Two chips in the same board (i.e. KW41 + Wi-Fi chip)

- External: KW41 design + 2.4 GHz environment (i.e. Wi-Fi Routers, 802.15.4 networks, Bluetooth devices)
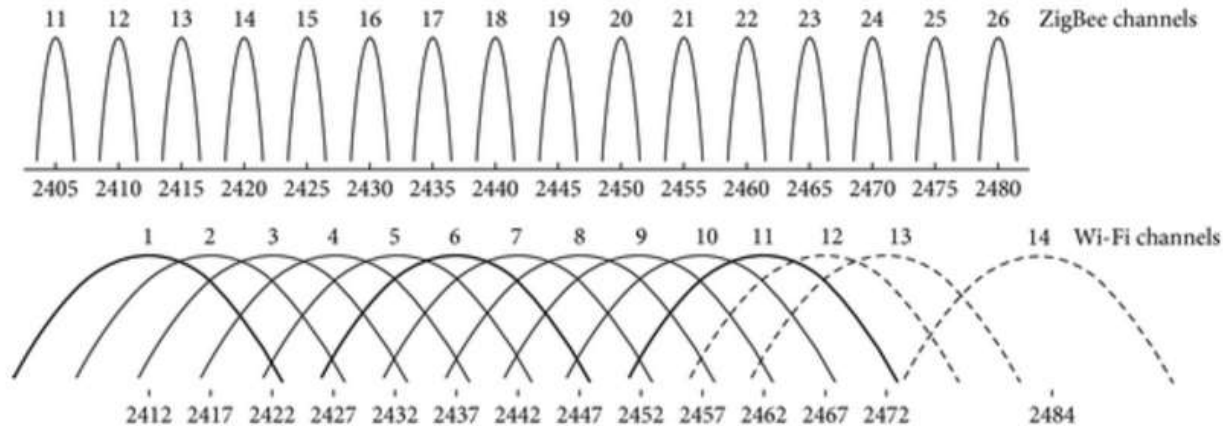
Wi-Fi + 802.15.4 Spectrum
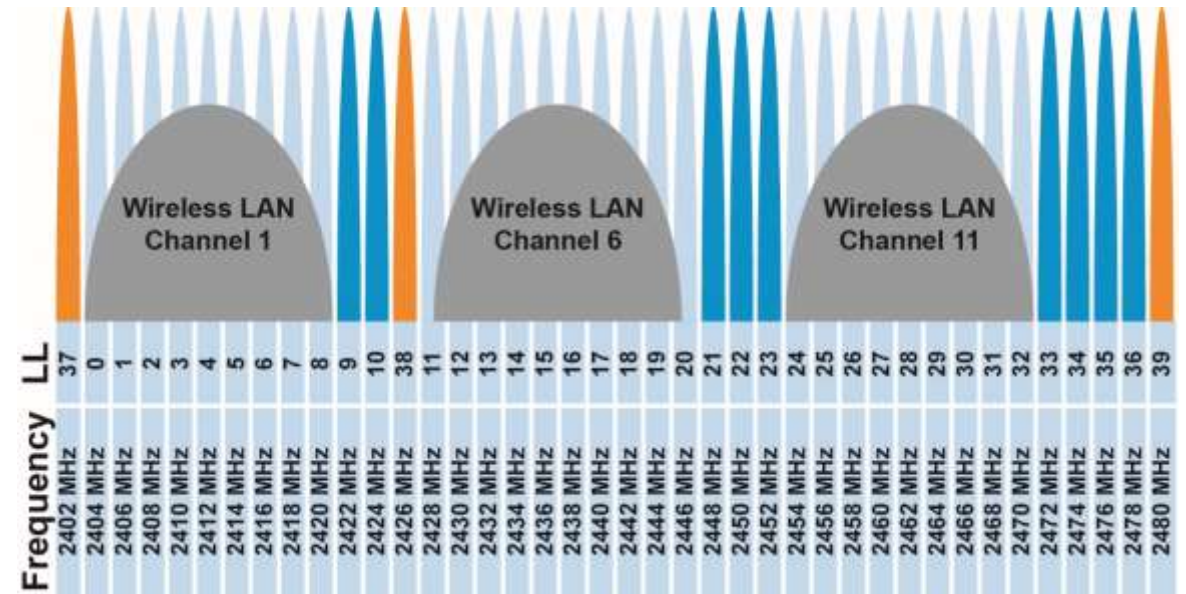
Wi-Fi + BLE Spectrum

Introduction (continue)

The KW41Z includes a 2.4 GHz transceiver that supports the following protocols:

- BLE
- 802.15.4 (Thread, ZigBee…)
- Generic FSK

Inter-ICs (off-chip) coexistence may be enabled by using three GPIOs to signal RF activity and request access to the medium.

Pin name	Direction	Description
RF_ACTIVE	Output	Signals when the transceiver becomes active
RF_STATUS	Output	Signals RF activity type (RX/TX) and the priority of the sequence
RF_DENY	Input	Signals if the access to the medium is granted or not

Mobile Wireless Systems Coexistence MWS

Is a set of APIs included in the Connectivity Framework

Allows Link Layers and higher layers control the access to the resources.

Allows inter-ICs coexistence (i.e. BLE + external Wi-Fi chip)
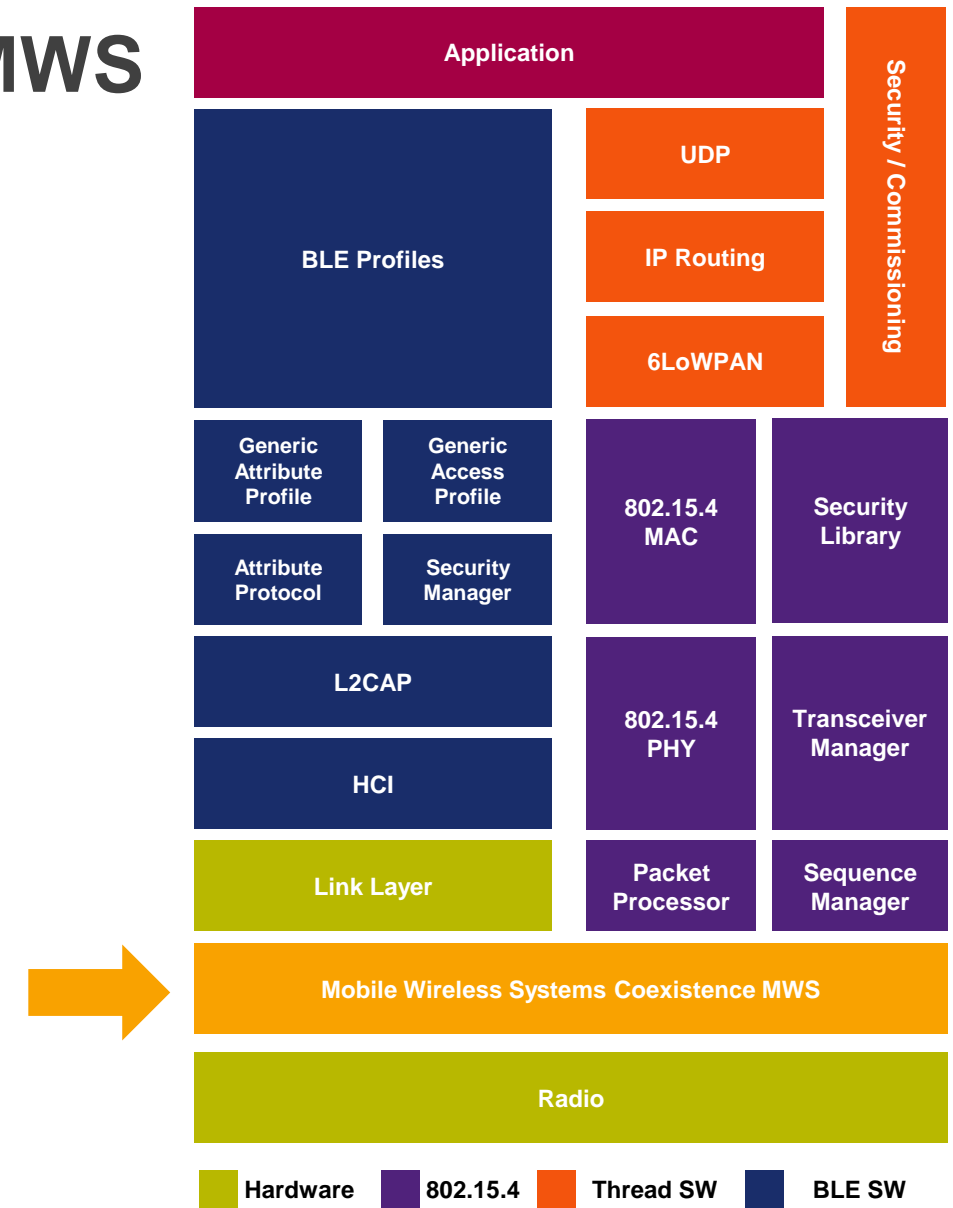
Thread + BLE KW41 Demo Application

BLE UART to Thread Shell

BLE Direct Communication

FRDM-KW41Z

Thread Network
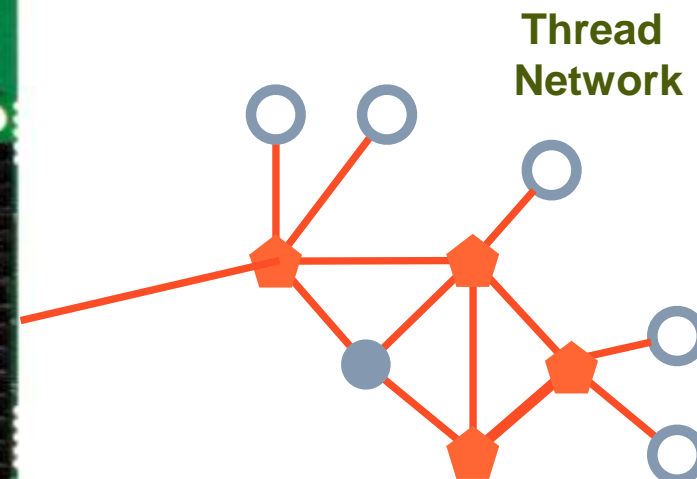
BLE+THREAD DEMO

HSDK

Host SDK (Host Software Development Kit)

Overview

The Kinetis Wireless Host SDK consists in a set of cross-platform C language libraries which can be integrated into a variety of user defined applications for interacting with Kinetis Wireless Microcontrollers.

- The Kinetis Wireless Host SDK is meant to run on Windows OS, Linux OS, Apple OS X ® and OpenWrt

- The HSDK software is designed to help developers interact with Host SDK from Python and C programming languages

Host SDK - THCI

The Host SDK implements two physical layers for transporting THCI:

- **UART - for direct UART**

The UART layer handles sending and receiving THCI frames over a serial interface to a Thread-enabled device.

C sample codes:

<installation path>NXP\MKW41Z_ConnSw_1.0.2\tools\wireless\host_sdk\hsdk\demo

Python sample codes:

<installation path>\MKW41Z_ConnSw_1.0.2\tools\wireless\host_sdk\hsdk-python\src\com\nxp\wireless_connectivity\test

Running script with THCI

PC

THCD* (KW41z)

*Thread Host Controlled Device

Executing script

The script "**getaddr.py**" sends THCI commands from the PC host to the board.

It performs a factory reset, creates a network and then it gets the device's **IPv6 addresses**

SNIFFING

Sniffer with Wireshark + USB-KW24D512

- Download and install the **Kinetis Protocol Analyzer Adapter** for USB-KW41Z or USB-KW24D512
- Direct link

- Support both Wireshark and Ubiqua.
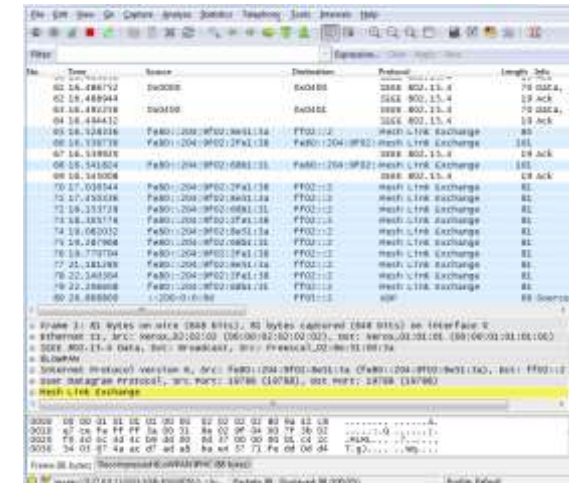
OVER THE AIR UPDATES

Over-The-Air (OTA) Updates

- For large networks, or for in the field updates, send new firmware wirelessly
- One server node sends the updated firmware, with one or more client nodes receiving.
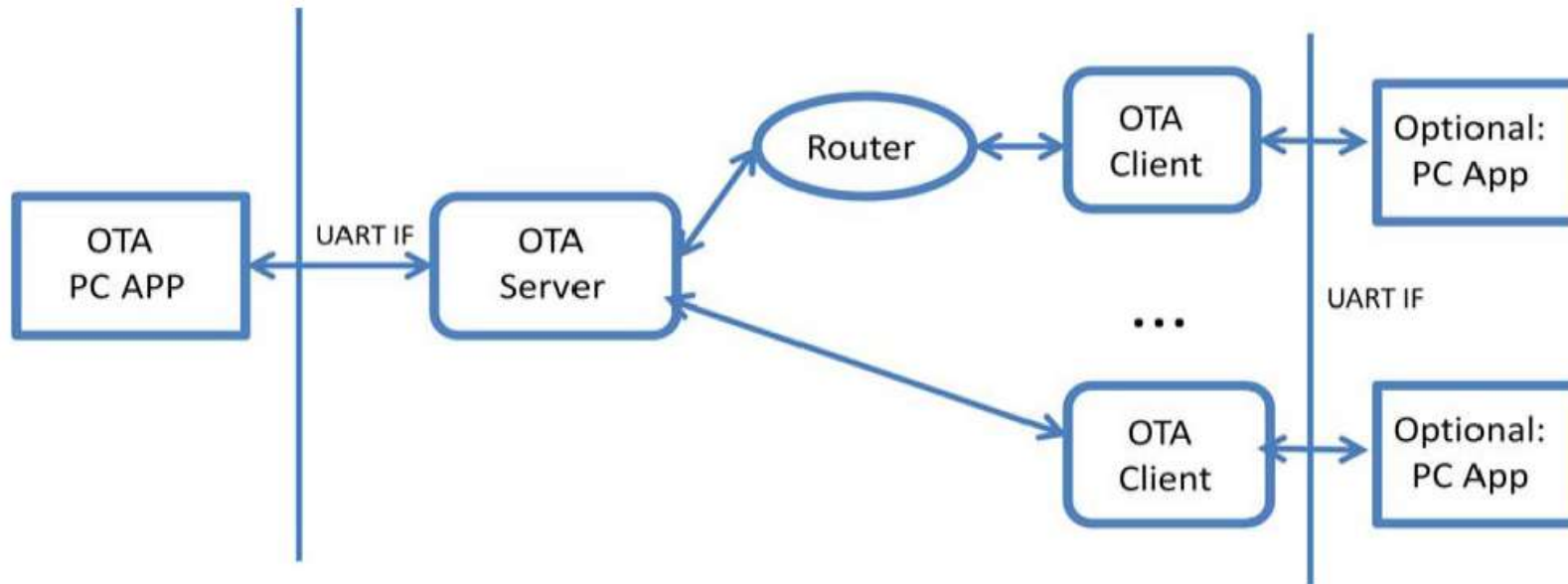- Uses Test Tool, a Windows program that communicates with the server node via FSCI (using a UART or USB interface to the board).

NXP Test Tool

- PC Software to communicate over serial paths to wireless development boards
  - FSCI Serial Communication for Host PC control and development prototyping
  - Used for Over-the-Air Update based Out-of-the-Box Examples
- Detailed instructions found in **\docs\wireless\Thread\Kinetis Thread Stack OTA Firmware Update User's Guide.pdf**
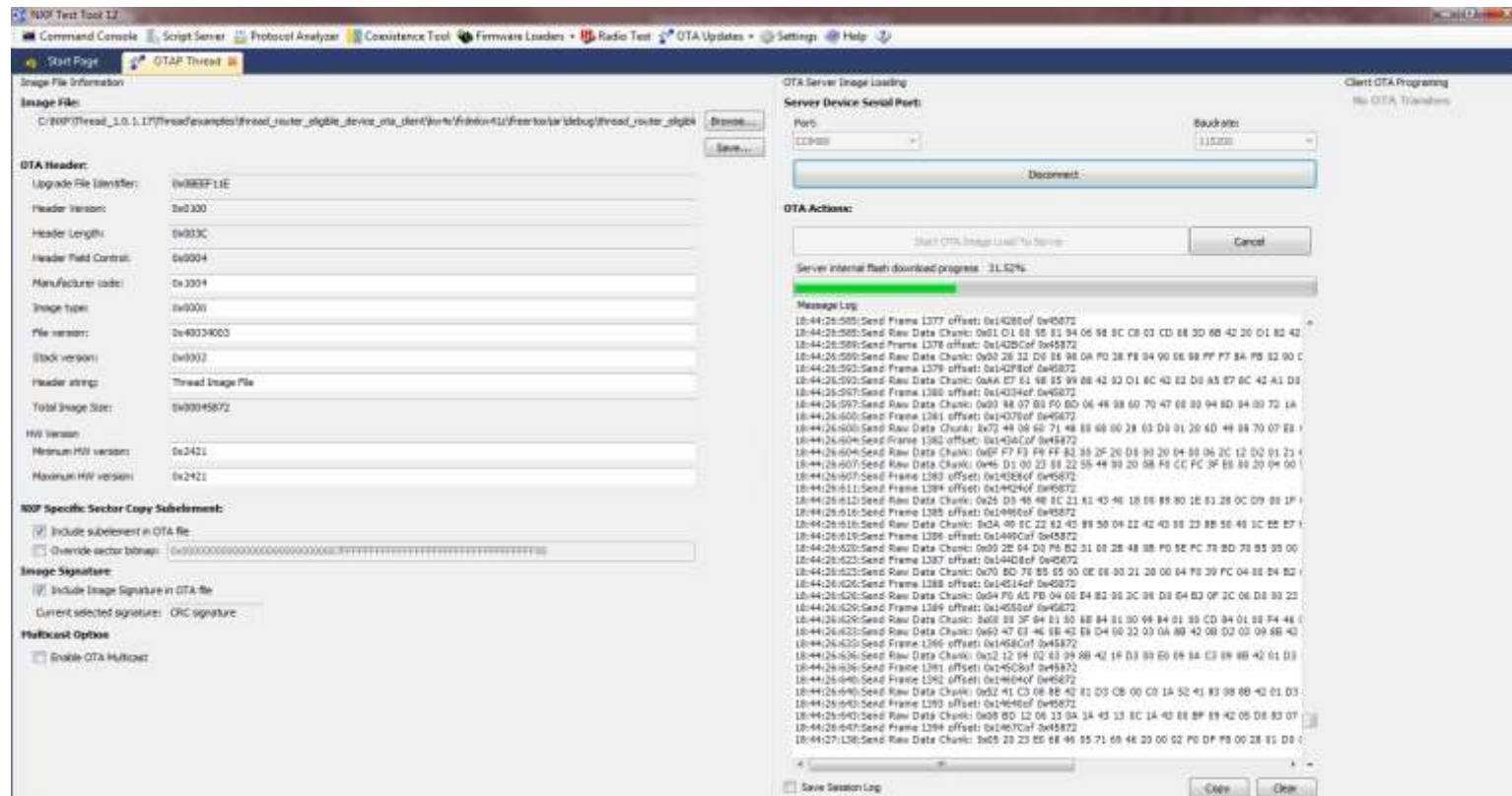
NXP Kinetis Wireless Solutions Agenda

- **KW41 Family Overview**

- **Bluetooth Low Energy (BLE)**

- **Thread**

- **BLE+Thread**

- **Wireless Framework**

- **Modular Gateway**

MODULAR IOT FRAMEWORK

Modular IoT Gateway: Optimized Combination of Technology

Control

MPU i.MX6UL

Connectivity

Thread Radio KW41Z
zigbee Radio JN5179
Wi-Fi 802.11b/g/n via 1DX
Bluetooth 4.1 via 1DX
Ethernet via i.MX6UL
2*USB OTG via i.MX6UL

Storage

4Gb LP-DDR3 RAM
8Gb NAND Flash
uSD Card

Security

Authentification A7001

Identification

NFC reader PN7120

Power

PMIC PF3001

Ethernet / 2*USB ports / Wi-Fi SMA antenna
USB debug / uSD card slot / 5VDC-3A

Wi-Fi
module

PN7120
Explorer Board
(NFC)

Kinetis KW41Z
Module on
Mezzanine (Thread)

JN5179
Module on
Mezzanine (zigbee)

i.MX6UL SOM
on App specific base board

Modular IoT Gateway | Multi-protocol interoperability

Pre-integrated, tested and certified southbound mesh support for the a wide array of wireless protocols, with flexibility to work together or independently, enabling end-to-end wireless communications in LNN configurations.
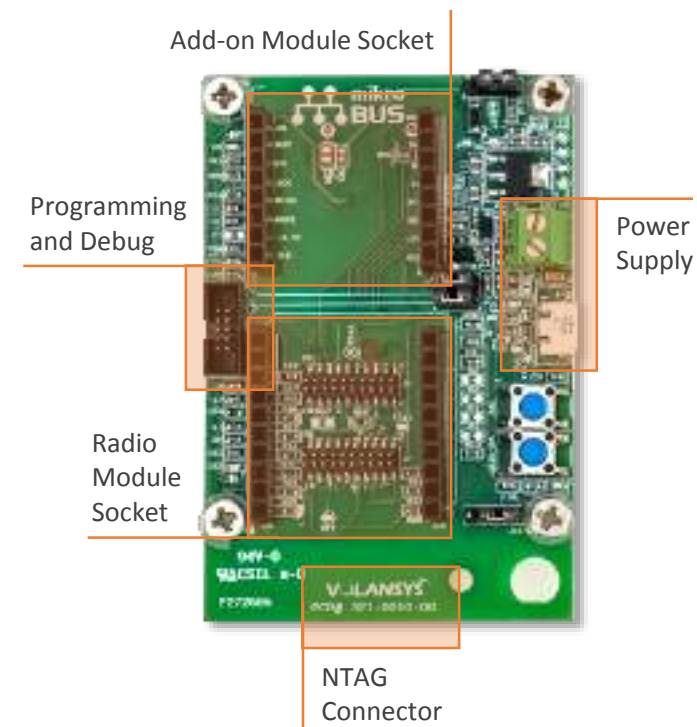
Southbound Mesh:

Northbound:

Application Layer Support
- ✓ CoAP per Thread Spec.
- ✓ CoAP Observe Funct.
- ✓ zigbee 3.0
- ✓ MQTT

NFC Tap & Connect Commissioning

Intrepid Smart App Commissioning

Currently Shipping Integrated Development Experience v1.0 !!

Modular Simple Edge Node: Platform and Modules

Add-on Module Socket

Programming and Debug

Radio Module Socket

Power Supply

NTAG Connector

**Shipping Today
with Modular IoT Gateway
Part# SLN-NTW-GTWY**

TAG
NFC/I2C

Hardware Modules

Radio Modules

KW2xD	KW41Z	JN5169	JN5179-001-M1x
Thread	**Thread**	**zigbee**	**zigbee**

Sensor/Actuator Add-on Modules

How To Engage with NXP IoT Solutions

…to begin your Thread Development:

- **Where to get more information [and to purchase the kit](#):**
  - www.nxp.com/modulargateway

- **Online Technical Documentation:**
  - www.nxp.com/go/modulargateway *(access required)*

- **Community Support for the Modular Framework:**
  - https://community.nxp.com/groups/modular-framework

NXP Kinetis Wireless Solutions Agenda

- **KW41 Family Overview**

- **Bluetooth Low Energy (BLE)**

- **Thread**

- **BLE+Thread**

- **Wireless Framework**

- **Modular Gateway**