THREAD

Thread Technical Overview
5 October 2015
Berlin

THREAD | What is Thread?

A secure, wireless mesh networking protocol that:

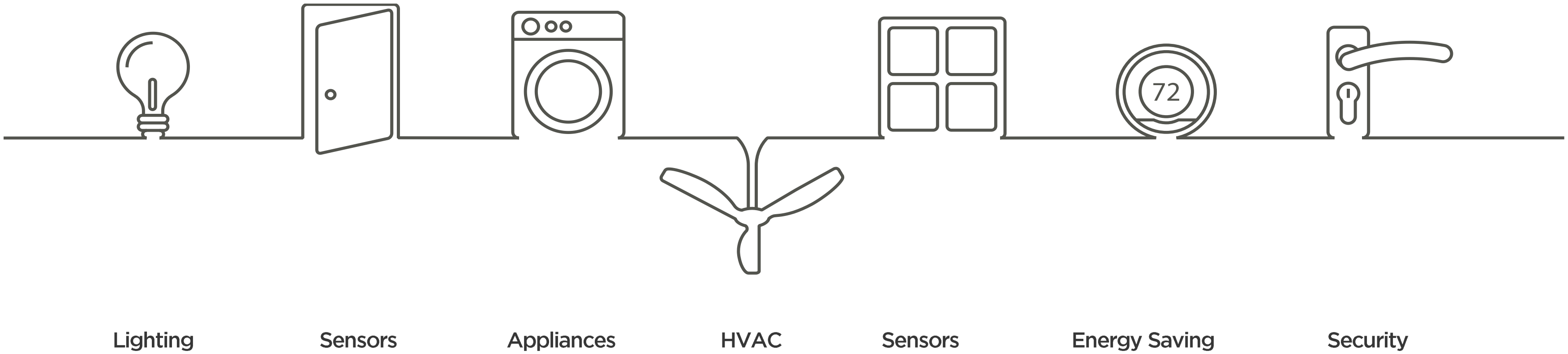
- Supports IPv6 addresses and simple IP bridging
- Is built upon a foundation of existing standards
- Is optimized for low-power / battery-backed operation
- Is intended for control and automation (250kbps)
- Can support networks of 250 nodes or greater
- Supports low latency (less than 100 milliseconds)
- Offers simplified security and commissioning
- Runs on existing 802.15.4 wireless SoCs

THREAD	Standard
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)

THREAD | Connected Home – Networking Requirements

- Direct addressability of devices – within the HAN and from smart phones or tablets
- Simplified forming and joining of network, limit special devices or customer knowledge of concepts like coordinator vs. router vs. end device
- Scalable to 200-300 devices in a home – with sufficient routers to provide coverage but remainder can be end devices
- Latency less than 100 milliseconds for typical interactions (user interaction concern)
- Allow the use of multiple gateways
- Seamless connectivity to user interaction on device of choice in the home (dedicated display, smart phone, tablet, etc.)
- Battery operated devices with years of expected life – door locks, security sensors etc.

THREAD | Target Devices



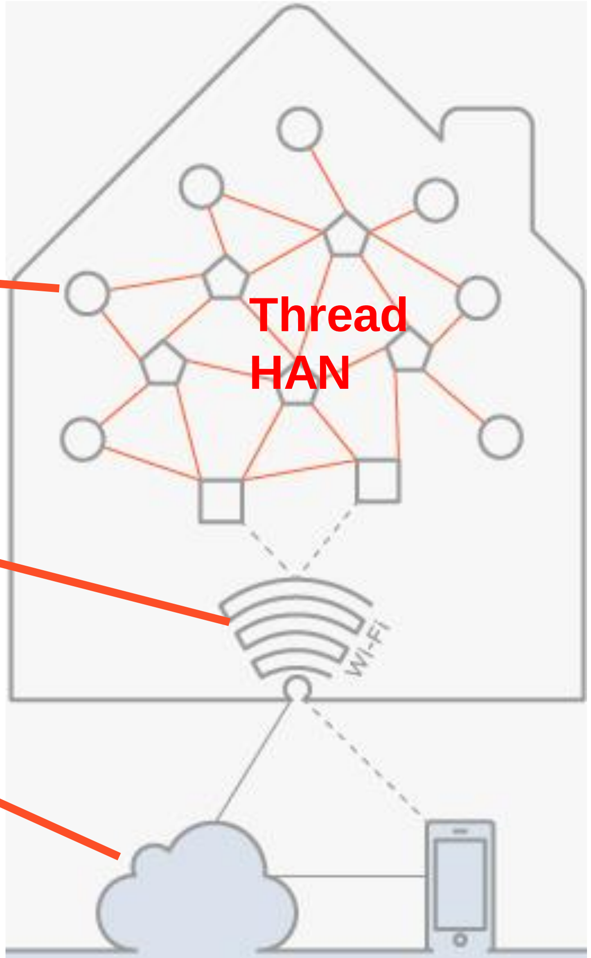
- Normally Powered
  - Gateway
  - Lighting
  - Appliances
  - Smart Meter
  - Garage door opener
  - HVAC equipment
  - Smart Plugs
  - Fans
- Powered or Battery
  - Thermostat
  - Light switches
  - Smoke detectors
  - In home display
  - Shades or blinds
  - Door bell
  - Glass break sensors
  - Robots/cleaners
- Normally Battery
  - Door sensors
  - Window sensors
  - Motion sensors
  - Door locks
  - Radiator valves
  - Body sensors

THREAD | System Messaging Model

- Expect device to device communication within HAN for operations in the home

- Border Router forwards data to cloud
- Also provides WiFi connectivity to phone or tablet in the home

- Cloud connectivity for control when not at home
- When within the home, phone or tablet must go direct to gateway to reduce latency
- Has to be seamless to consumer using device



THREAD | Key Features Overview

IP-based:

- Simplified bridging to other IP networks

Flexible Network:

- Simplified device types

Robust:

- No single point of failure

Secure:

- Simple security and commissioning

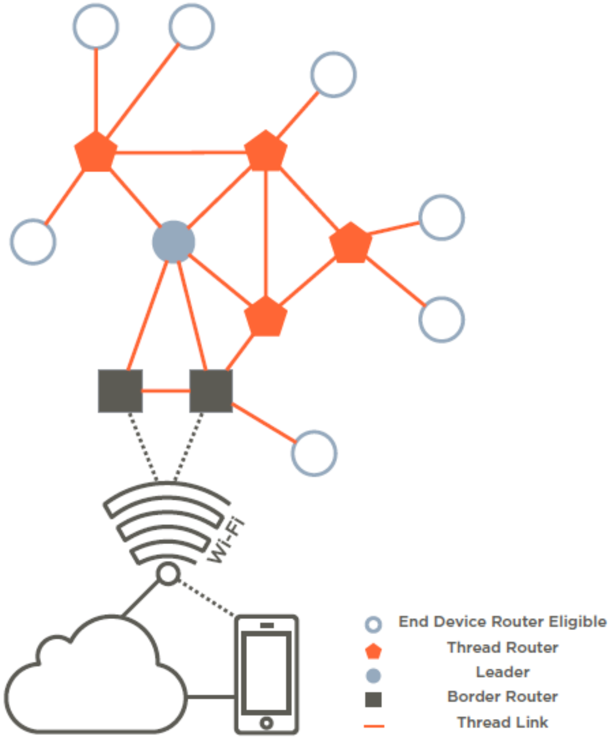
Low Power Operation:

- Support for sleeping devices

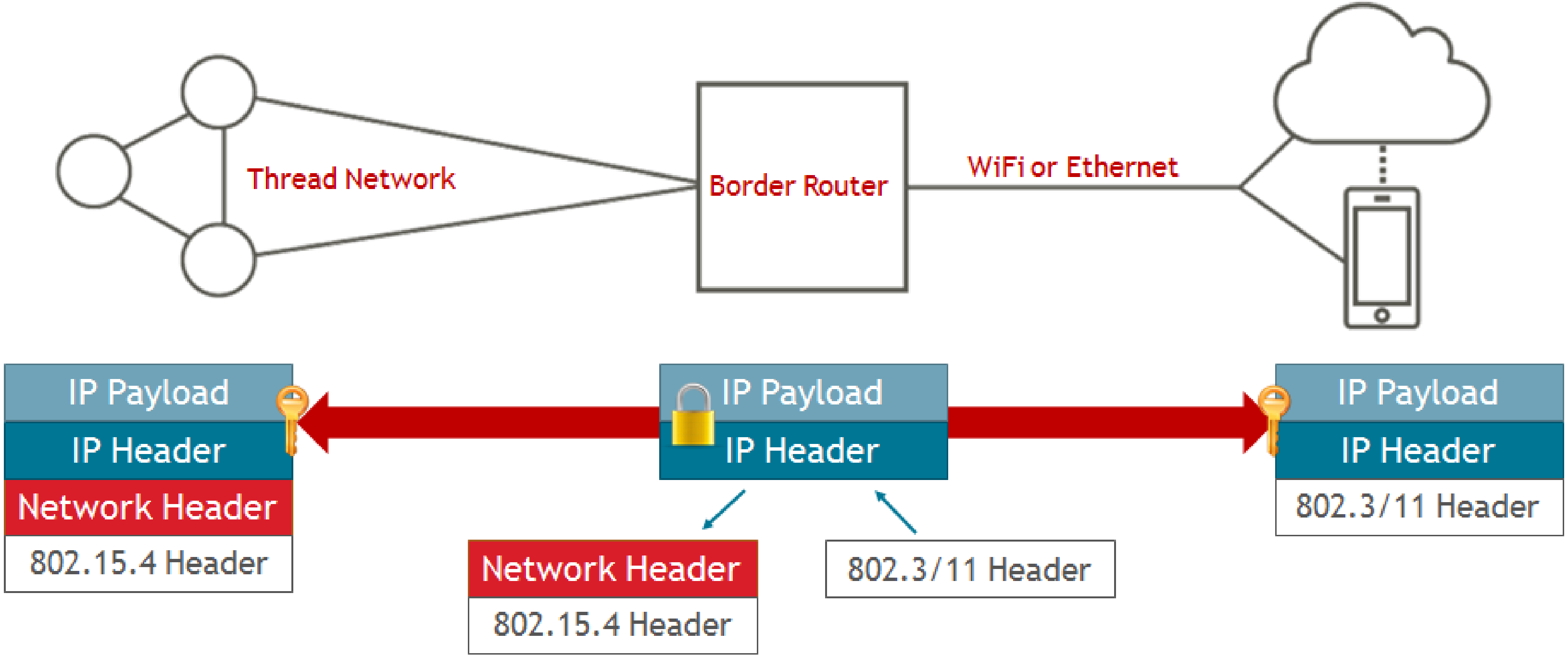
Thread	Standard
Application Layer	
UDP + DTLS	RFC 768, RFC 6347, RFC 4279, RFC 4492, RFC 3315, RFC 5007
Distance Vector Routing	RFC 1058, RFC 2080
IPv6	RFC 4944, RFC 4862, RFC 6282, RFC 6775
6LowPAN	
IEEE 802.15.4 MAC (including MAC security)	IEEE 802.15.4 (2006)
Physical Radio (PHY)	

THREAD | IP-Based: Direct Addressability

- All devices have IPv6 address plus short address on HAN
- DHCPv6 used for router address assignment
- Home Network can directly address devices through Border Routers
- Cloud Services can address devices from the Internet
- Devices can address local devices on HAN or off network devices using normal IP addressing

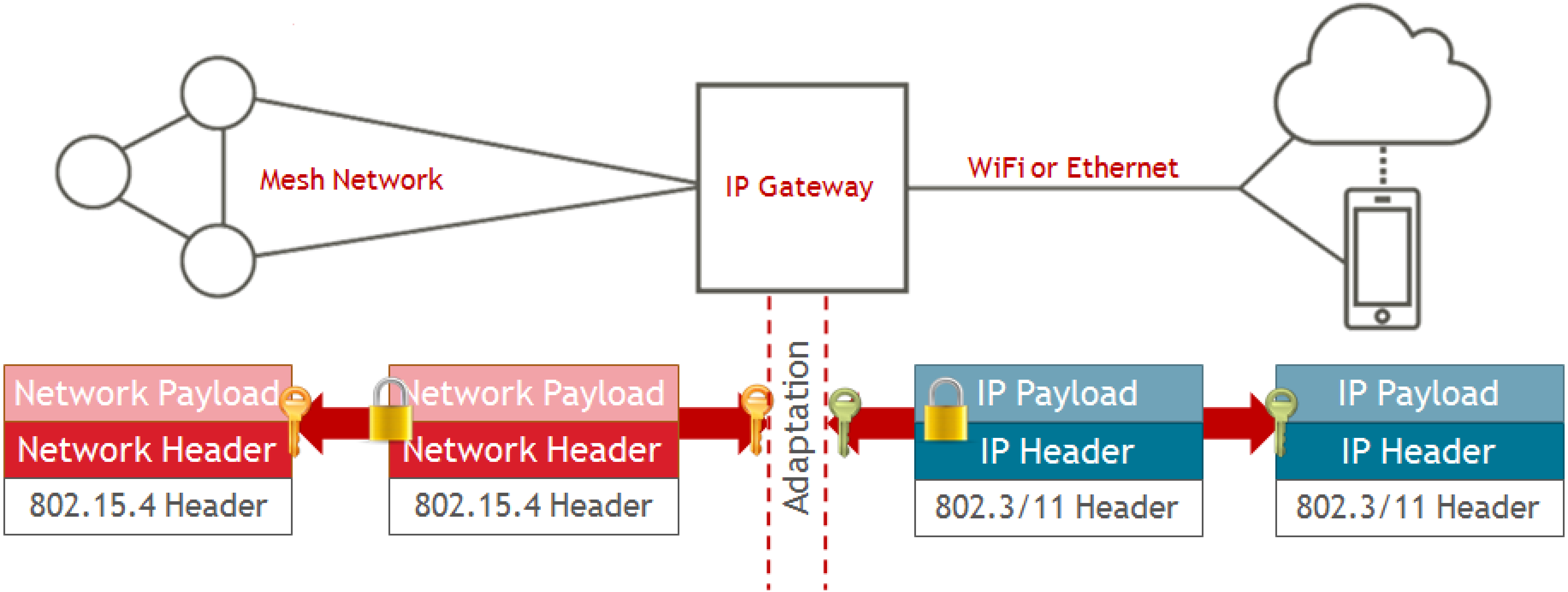


THREAD | **IP-Based: Simplified IP Bridging**



1. Simplified bridging between mesh network and Internet
2. Enables end-to-end IP security

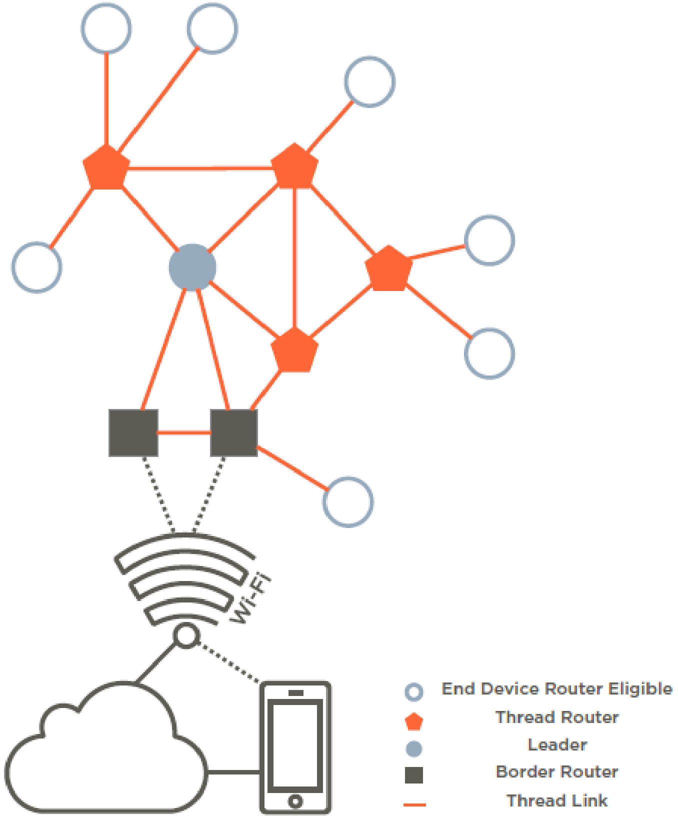
THREAD | IP-Based: Comparison to non-IP mesh networks



1. Network header (and network addresses) must be adapted to IP
2. Payload re-secured at IP Gateway and may require some adaptation for IP

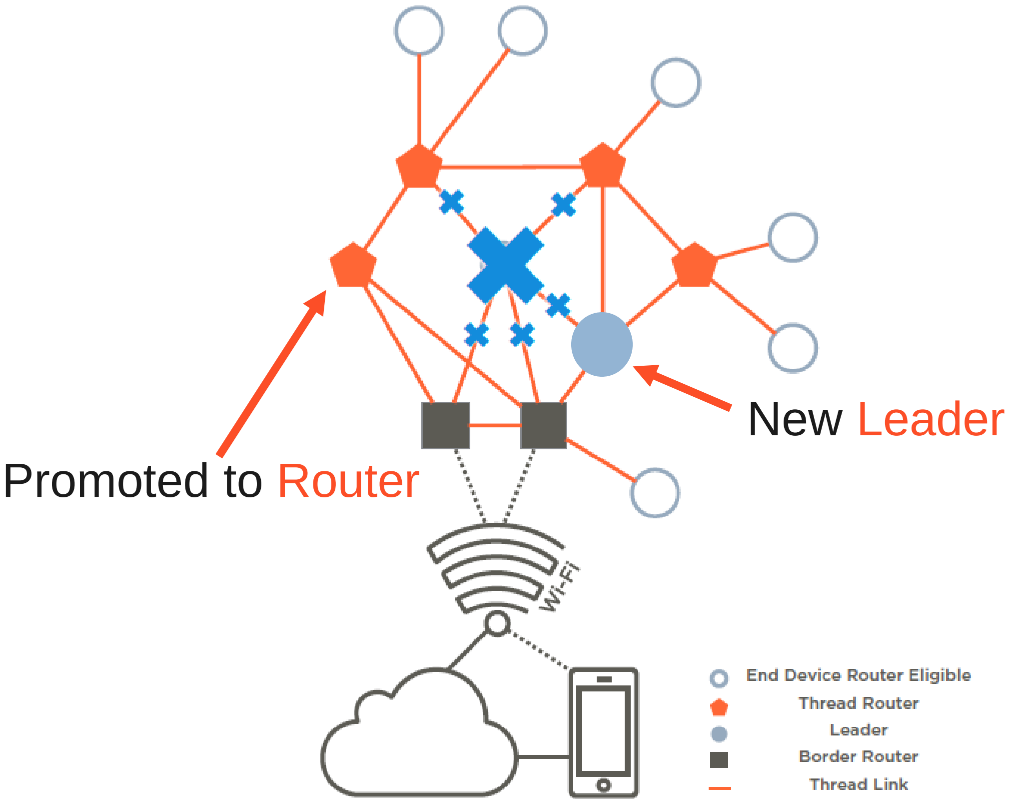
THREAD | Flexible: Simplified Device Types

- Devices join as Router Eligible or End Device
- Router Eligible: Can become Routers if needed
  - First router on network becomes Leader
  - Leader: Makes decisions within network
- End Devices: Route through parent
  - Can be “sleepy” to reduce power consumption



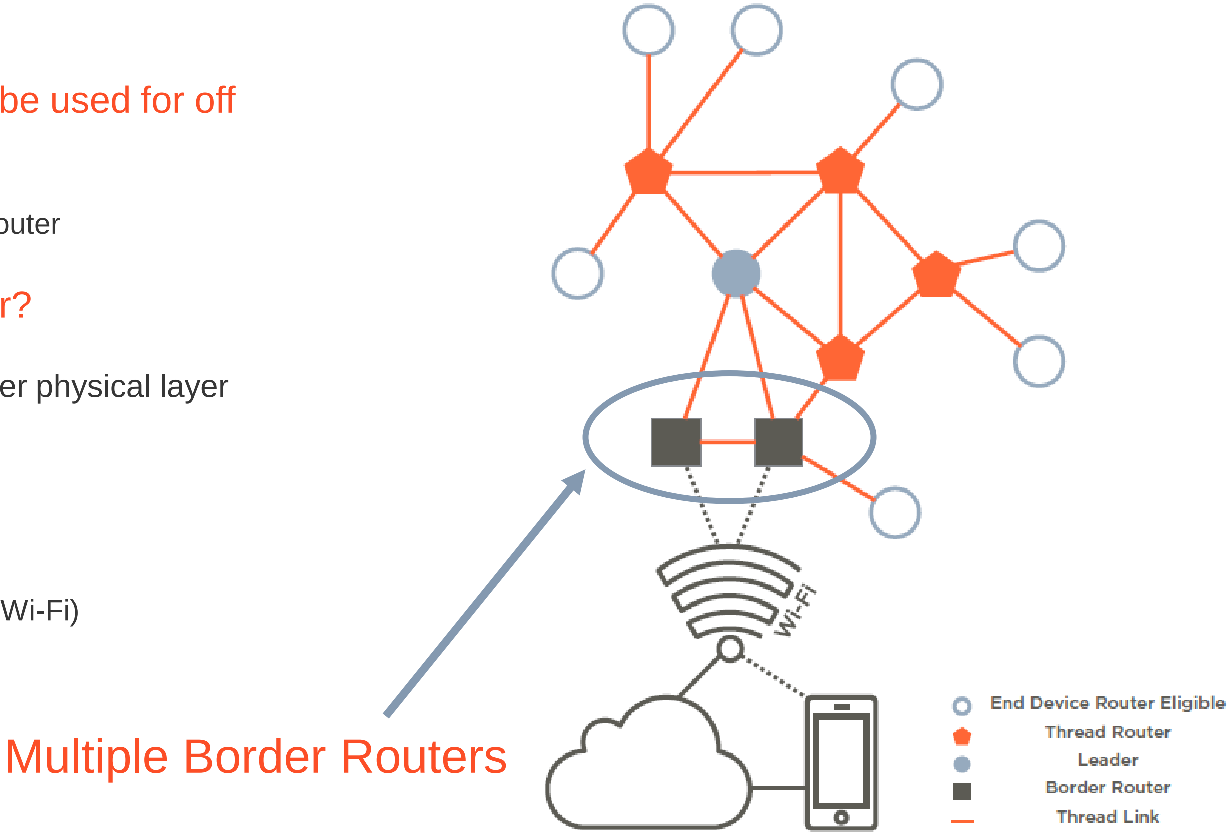
THREAD | Robust: No Single Point of Failure

- Dynamic Leaders
  - If Leader fails, another Router will become Leader’
- Router Promotion
  - Leader can promote Router Eligible devices to Routers to improve connectivity if required



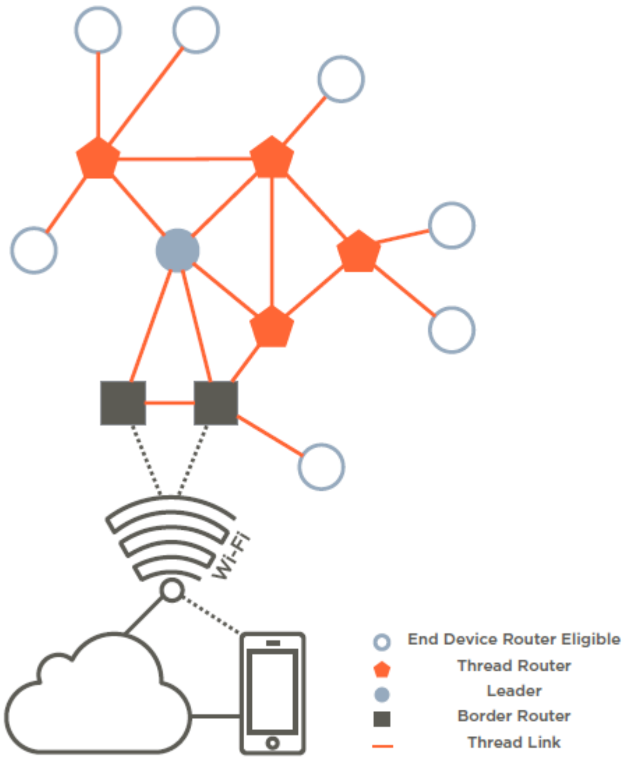
THREAD | Robust: No Single Point of Failure

- Multiple Border Routers can be used for off network access
  - Devices operate without Border Router
- What can be a Border Router?
  - Anything with 15.4 chip and other physical layer
    - Home Wi-Fi router
    - Set top box
    - Smart Thermostat (15.4 and Wi-Fi)



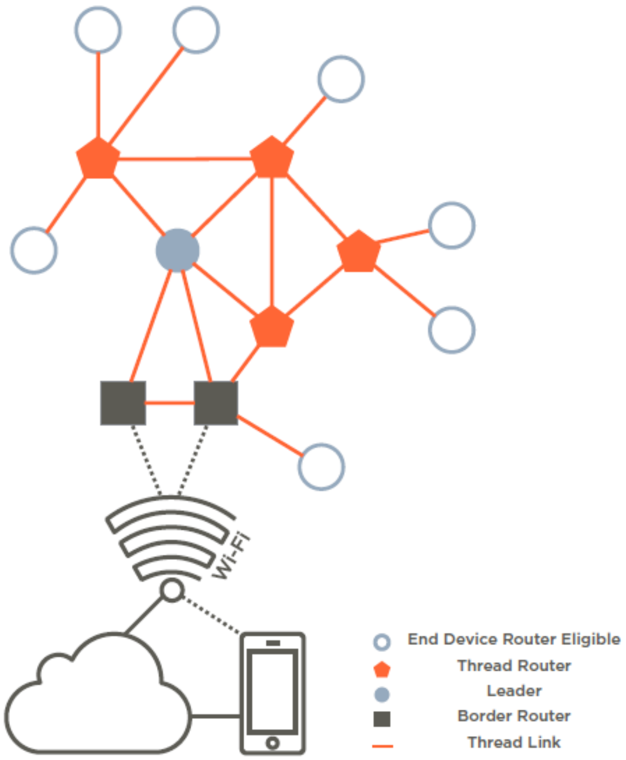
THREAD | Secure: Security and Commissioning

- Simple Commissioning
  - User authorizes devices onto the network using smart phone or web
  - Can be done on network if there is a device with a GUI
- DTLS Security session established between new device and commissioning device to authenticate and provide credentials
- Once commissioning session is done – device attaches to network
- MAC security used for all messages
- Application level security used based on end-device requirements

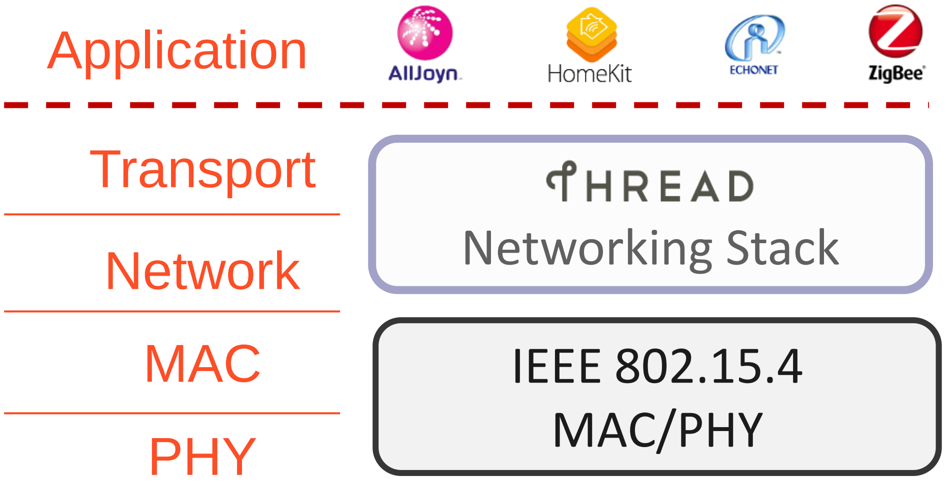


THREAD | Lower Power Operation: Sleepy Devices

- Sleeping devices poll parents for messages (or remote device if application configured)
- Sleeping device not required to check in to allow lower power operation
- Parents hold messages for sleeping devices
- Sleeping device automatically switches parent if it loses connectivity



THREAD | The WiFi of Mesh Networking



- Thread defines how data is sent in network but not how to interpret it
  - Low-power, mesh networking equivalent to WiFi
- Thread can support IP-based application layers, but does not define one
  - Provides basic services such as: UDP messaging and acknowledge, Multicast messaging
  - ‘Application layers not using IP services would need some translation / adaptation

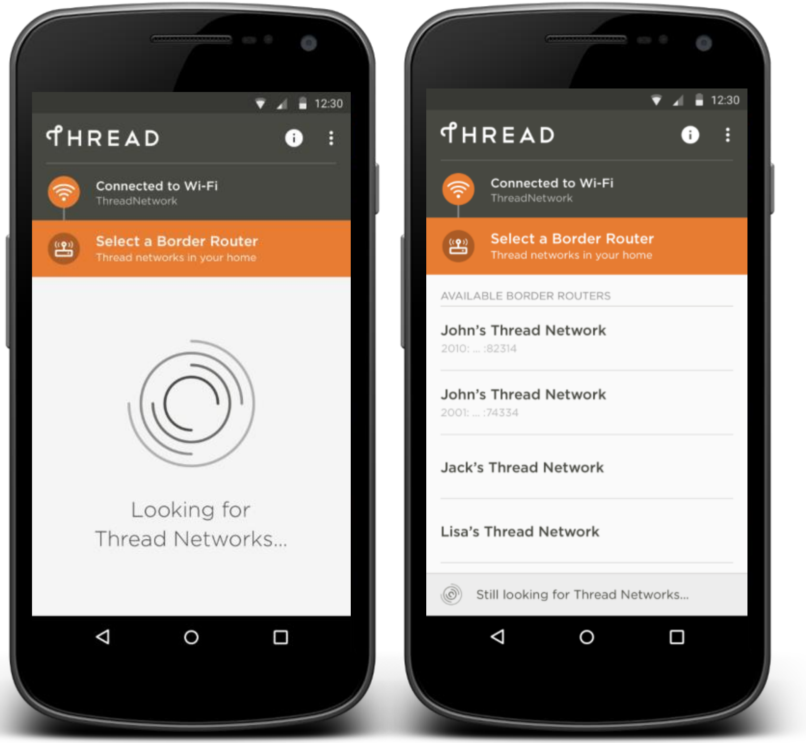
THREAD | Certification

- All Thread devices will require network certification
- Validate device behavior
  - Commissioning
  - Network functionality
  - Device operation in network
- Members will have access to free standard test harness
- Certification through a 3rd party test lab (UL)
  - Certification program to launch in October 2015

CONNECTS WITH THREAD

THREAD | Commissioning Application

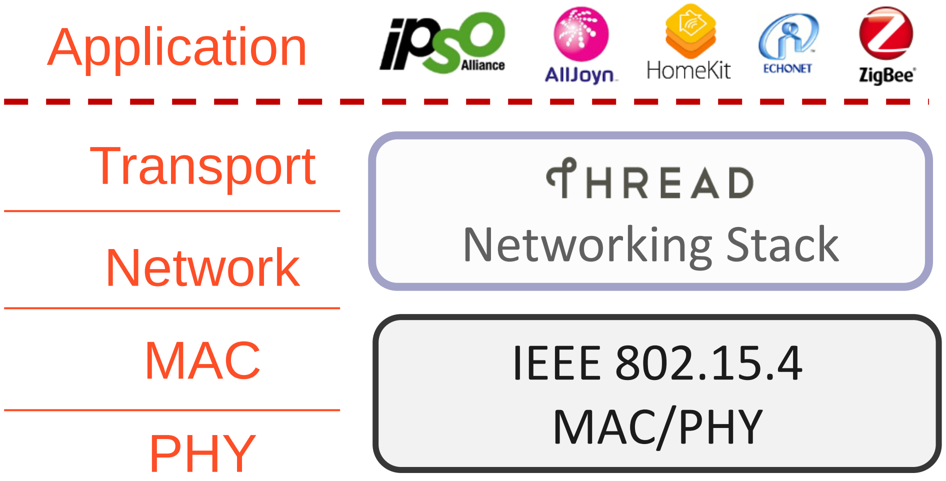
- Enables users to effortlessly add devices on the Thread Network and manage settings
- Designed as a reference app with source code for all Sponsors and Contributors
- Communication via Thread Group's MeshCoP Protocol and COAP
- Communication library written in C and C++, leveraging code that is extendable across iOS and Android.



THREAD | Current Status of Thread

- Thread specification released to Members July 14
- Thread test harness expected to be released to Members in October
- Thread also releasing a commissioning application built to Members in October
- Silicon Labs Thread stack released in July
- Update to Network Analyzer and AppBuilder
- Also released Border Router reference design software

THREAD | The WiFi of Mesh Networking



- Thread defines how data is sent in network but not how to interpret it
  - Low-power, mesh networking equivalent to WiFi
- Different customers interest in different application layers
  - Market is going to help resolve this

Thread Stack

THREAD | Routing

- Simple distance vector algorithm
- Provides next-hop information about all router nodes
- Highly compressed protocol format: one byte per destination
- No reactive route discovery by devices
- Child ID encodes parent router ID so route is known when address is known
- Point to point routes always available to every router

THREAD | Router Selection

- Limit of 32 active routers to reduce bandwidth and RAM consumption
    - 64 router addresses to allow timing out and reassignment
- No neighbor selection required
- Routers select automatically from router eligible end devices (REED)
- REED behave as end devices, but listen to routing messages

- As number of routers increases, routers can elect to become REED
- If REED notes need to become a router it will petition leader

THREAD | Leader

- Decision maker in network, chosen autonomously
- Assigns router ID’s
- Assigns 6LoWPAN contexts
- Collates border router information
- Assembled network data is distributed using Trickle/MLE advertisements
- All routers store the network data, only the leader can make changes to it

THREAD | No single point of failure

- Recovery from leader failure or disconnected topology by self election of new leader
- Network fragments automatically elect a new leader, and if reconnected the leader returns to being a router

THREAD | Network Data

- All volatile data in one packet
  - Leader identity (node ID and EUI64)
  - Network data sequence number
  - Link data for all routers
    - Two bit incoming link quality
    - Two bit outgoing link quality
  - Routing data for all routers
    - Four bit routing cost
    - Five bit next hop identity
- Stable network data can also be sent
  - Context ID’s
  - Border router information

THREAD | 6LoWPAN Header Compression

- IPv6 addresses and headers are 16 and 40 bytes long respectively
- Compression achieved by eliding known fields including 8 byte prefixes
- Thread uses 6 LoWPAN mesh headers for next hop forwarding
- IETF RFC 4944 and RFC 6282

THREAD | IPv6 Addresses

- IPv6 address is 64 bit prefix ++ 64 bit interface ID
- First few bits of prefix indicate unicast, multicast etc
- For unicast, 64 bit prefix identifies a subnet, 64 bit interface ID identifies a device on that subnet
- Devices have multiple unique addresses
- Written like FE80::00FF:FE00:12AB, colons separate words, double colon elides a block of zeros
- /x is a prefix length: FE80::/64 is a 64 bit prefix consisting of FE80 followed by 48 zeros

THREAD | Lots of IPv6 Addresses

- link local (one hop only)
  - FE80::/64 where IID is modified EUI64 (LL64 address)
  - FE80::00FF:FE00:xxxx where xxxx is the MAC 16-bit address (LL16 address)
- ULA (Unique Local Address - nonroutable subnet ID)
  - one fixed, permanent ULA for the subnet
  - same two IIDs as link local (ULA64 and ULA16 addresses)
- globally routable
  - managed by border routers (GP64 addresses)
  - no IIDs derived from MAC 16-bit address
  - for security and privacy not using modified EUI64 address either
- IPv6 addresses derived from MAC 16-bit addresses are not used at the application layer

THREAD | Finding 16 bit MAC ID for Destination

- Address cache maps IPv6 addresses to MAC IDs
- ULA16 → done
- In cache → done
- ULA64 → Send multicast query to routers
- GP64 → unicast query to appropriate DHCPv6 server
- Other (off mesh) → pick appropriate border router

THREAD | No explicit ND Messages

- IPv6 uses Neighbor Discovery for auto-configuration
- For Thread – ND is merged into other protocols
- No DAD (duplicate address detection) for LL64 addresses
- ID assignment done using Thread Management Protocol (CoAP)
- Neighbor and router discovery are done using MLE advertisements

THREAD | DHCPv6

- different DHCPv6 servers handle different prefixes
- nodes maintain multiple DHCPv6 clients
- router has (effectively) a /117 prefix, the /112 prefix FE80::0000:00FF:FE00:... followed by its own router ID
- distinguished leader node has /112 prefix given out as /117 prefixes to routers
- border router has /64 from obtained from outside

THREAD | Border Routers

- send data to leader
  - prefix
  - if is default router
  - if has DHCPv6 server

- leader includes this data in network data, adding border router’s 16-bit ID and a 6LoWPAN context ID

- outgoing messages are tunneled using mesh headers to an appropriate border router

- incoming messages are tunneled to their destination

THREAD | 16-bit node ID assignment

- folded into MLE to reduce message overhead
- all nodes join as end devices, can then request router IDs
- parent passes node ID to child as part of MLE handshake (saves two messages)
- ID timeout is same as MLE child link timeout (saves DHCP renewal messages)

THREAD | MAC and MLE Key Distribution

- All messages secure using either MAC key (network wide) or MLE key
- MAC and MLE keys are an HMAC hash of 32-bit extended key identifier using a master key
- low-order byte of extended key identifier is used as 802.15.4 key identifier
- unicast master key distribution is used only when one or more devices need to be kicked out

Thread Commissioning

THREAD | Commissioning Model

- Devices must be securely authorized onto the Thread network by a user

- Can be done with a variety of devices
  - On network using a device with a GUI
  - On local Home network using border router
  - To the web using border router

- User must enter device passphrase which is used to authenticate device onto the network

THREAD | Basic steps in Commissioning

- Two separate Authentications required:
  - Commissioning device authenticated as Active Commissioner – allowed to add devices to the network
  - Joining device is then authenticated by Active Commissioner – then device is provided network and security material to attach to the network
- Commissioning device is not provided network or security credentials due to security concern of having this material off network in devices

THREAD | Commissioning – Authorizing the Commissioner

- On network start up a commissioning passphrase is selected that is then used by commissioning devices to authenticate to the border router
  - User then has choice of providing this passphrase to other devices to allow them to commission
  - User can change this passphrase to eliminate other commissioning devices

- Commissioning device (off network) establishes a secure session (DTLS) with the border router using a commissioning passphrase (configured as initiation of border router and can be transferred between commissioning devices) using the commissioning passphrase

- Border router request commissioning session from leader
  - To ensure only one commissioner active at a time in the network

- Leader notifies network that a commissioner is active

THREAD | Commissioning – Joining a Device

- Joining device looks for network that is actively commissioning and finds router on that network (Joiner router)
- Joiner router acts as security point and relays messages from joiner to commissioner
- Joining device and Commissioner establish DTLS session using devices short passphrase
- When device is authorized by commissioner, the joiner router is notified that it can provide network credentials to joining device
  - Commissioning does not have network and security material (to reduce security risk)
  - Credentials sent to joining device encrypted with key established during commissioning authorization and sent to joiner and joiner router
- Device can then attach to the network

THREAD | Protocols Used in Commissioning

- DTLS sessions used between devices (requires use of DTLS 1.2)
- J-PAKE based in elliptic curve using NIST P-256 curve

- These libraries are required on joining device, border router and commissioner during the process

THREAD | Other Commissioning Notes

- Thread is working to build sample commissioning application for member companies to use and extend (for smart phone/tablet)
- Stack companies need to provide sample border router implementation for use with this commissioning application

- Commissioning is supported in the current Silicon Labs drop but doing it on-network and not through the Border Router

Thread Application Example

THREAD | Thread App

- Thread app is our test app.
- It contains all functionality and commands necessary for a node to operate on a thread network.
- It offers a CLI interface and a set of commands
- It is available on port 4901

THREAD | Thread App Bring Up

- We'll cover three main topics

  1. Forming, commissioning and joining
  2. Pinging
  3. How the sample application sends application data

THREAD | Forming

- Forming is performed via the "form" command. The last argument is optional.
  - 0: channel OR 0 for all channel mask, int8u
  - 1: power, int8s between -19 (low) and 3 (high)
  - 2: node type 2=router, 3=end device, 4=sleepy end device
  - 3: network ID, string
  - 4: ULA - optional, prefix string, like "FD01::"

- Example

  form 11 3 2 "net-id" "FD01::"

THREAD | Commissioning

- The would-be commissioner first petitions to gain commissioning status
  com_petition "please"

- If the petition is successful, the CLI prints:
  am commissioner, joining disabled

- The commissioner then sets the join key
  set_join_key "1JMRYXP7"

- The CLI then prints
  “am commissioner, joining enabled”

THREAD | Joining

- Joining is performed via the "join" command with arguments:
  - 0: channel, int8u
  - 1: power, int8s between -19 (low) and 3 (high)
  - 2: node type, 2=router, 3=end device, 4=sleepy end device
  - 3: network ID, string
  - 4: extended pan ID, array of bytes (may be an empty)
  - 5: pan ID, int16u (may be 0xFFFF)
  - 6: join key, 8-character string

- Example

  join 11 3 2 "net-id" {} 0xFFFF "1JMRYXP7"

THREAD | Pinging

- Nodes ping via the **ping** command
- Pick two nodes, A and B
- Find the IP address of A via **network_state**
  default ip: FD01:0000:0000:0000:0200:0000:000A:0005
- Issue the **ping** command on node B
  ping “FD01:0000:0000:0000:0200:0000:000A:0005”
- If successful
  - node A will display **ICMP ECHO_REQUEST**
  - node B will display **ICMP ECHO_REPLY**

THREAD | Sending Application Data

- CoAP is an application-layer transfer protocol that thread uses
- It can be thought of as binary HTTP
- It runs over UDP
- It has guaranteed delivery via resends and ACKs
- It guarantees in-order message delivery via message IDs
- Has four methods that are similar in function and utility to those in HTTP:
    - GET
    - POST
    - PUT
    - DELETE

THREAD | CoAP Message Types

- Messages can be either
  - confirmable (CON)
  - non-confirmable (NON)

- Confirmable messages require an ACK, while non-confirmable messages don’t

- If we don’t need reliability, we use NON
  - For example, a sensor broadcasting data

- If we need reliability, we use CON
  - For example, issuing a GET to a server

Confirmed, Piggybacked GET Example

```
Client                   Server     Client                   Server
   |                        |          |                        |
   |   CON [0xbc90]         |          |   CON [0xbc91]         |
   | GET /temperature       |          | GET /temperature       |
   |   (Token 0x71)         |          |   (Token 0x72)         |
   +----------------------->|          +----------------------->|
   |                        |          |                        |
   |   ACK [0xbc90]         |          |   ACK [0xbc91]         |
   |   2.05 Content         |          |  4.04 Not Found        |
   |   (Token 0x71)         |          |   (Token 0x72)         |
   |     "22.5 C"           |          |   "Not found"          |
   |<-----------------------+          |<-----------------------+
   |                        |          |                        |
```

Figure 4: Two GET Requests with Piggybacked Responses

THREAD | GET Request with a Separate Response

```
Client              Server
   |                  |
   |   CON [0x7a10]   |
   | GET /temperature |
   |   (Token 0x73)   |
   +----------------->|
   |                  |
   |   ACK [0x7a10]   |
   |<-----------------+
   |                  |
   ... Time Passes  ...
   |                  |
   |   CON [0x23bb]   |
   |   2.05 Content   |
   |   (Token 0x73)   |
   |     "22.5 C"     |
   |<-----------------+
   |                  |
   |   ACK [0x23bb]   |
   +----------------->|
   |                  |
```

Figure 5: A GET Request with a Separate Response

THREAD | Non-Confirmable GET with Non-Confirmable Response

```
Client              Server
   |                  |
   |   NON [0x7a11]   |
   | GET /temperature |
   |   (Token 0x74)   |
   +----------------->|
   |                  |
   |   NON [0x23bc]   |
   |   2.05 Content   |
   |   (Token 0x74)   |
   |     "22.5 C"     |
   |<-----------------+
   |                  |
```

Figure 6: A Request and a Response Carried in Non-confirmable Messages

THREAD | CoAP and the Sample App

- The sink sends a CoAP POST broadcast to identify itself
- The sensor hears the POST and stores the sink's IP address
- The sensor then sends its sensor data via CoAP POSTs that are unicasted to the sink's IP address