CSE 291: Wireless and Communication in the Internet of Things

Thread

Pat Pannuto, UC San Diego

ppannuto@ucsd.edu

Today's Goals

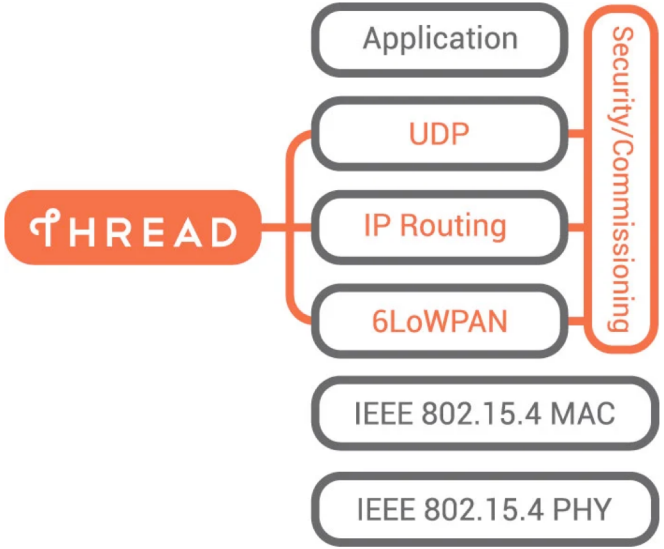
- Dive deep on Thread

Outline

- Thread
  - Overview
  - 6LoWPAN
  - Addressing & Routing
  - Runtime

Thread overview

- Build a networking layer on top of 15.4
  - Reuses most of PHY and MAC
  - Adds IP communication
  - Handles addressing and mesh maintenance
- Goals
  - Simplicity – easy to install and operate
  - Efficiency – years of operation on batteries
  - Scalability – hundreds of devices in a network
  - Security – authenticated and encrypted communication
  - Reliability – mesh networking without single point of failure
- Industry-focused, but based in academic research



References on Thread

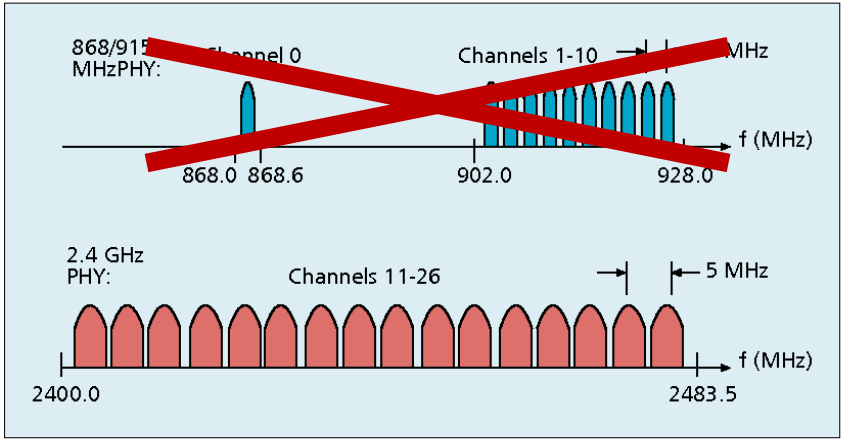
- Request for specification: https://www.threadgroup.org/ThreadSpec
  - Frustratingly locked down 😡

- Overview on capabilities: https://openthread.io/guides/thread-primer
  - Excellent overview
  - Lifting heavily for these slides

Changes to Physical Layer

- Remove all non-2.4 GHz PHY options

- Otherwise the same
  - OQPSK
  - 16 channels, 5 MHz spacing
  - Typical TX power 0 dBm
  - Typical RX sensitivity -100 dBm



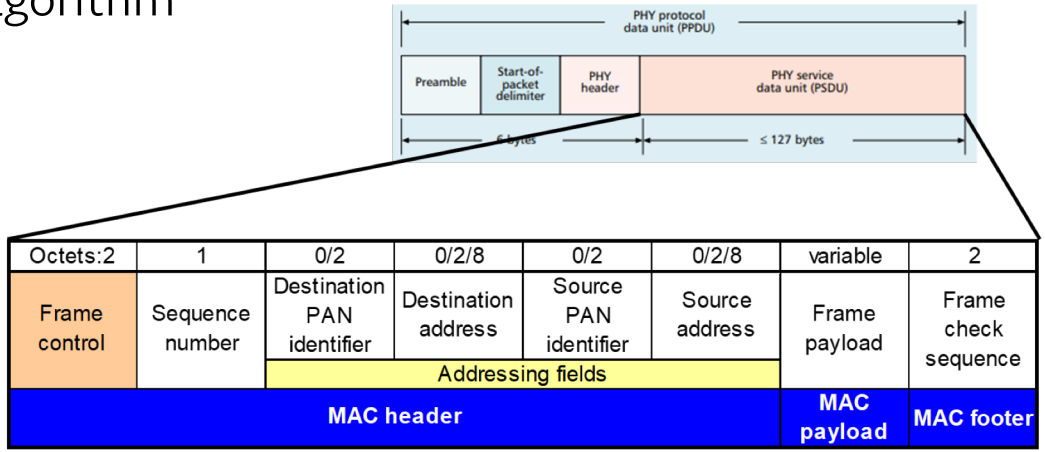
Changes to Link Layer and MAC

- Non-beacon-enabled PAN only
  - No superframe structure
  - No periodic beacons
  - No Guaranteed Time Slots

- Throw out most existing MAC Commands
  - Remove network joining/leaving
  - Remove changing coordinators
  - Remove Guaranteed Time Slot request
  - Network joining will be handled at a higher layer

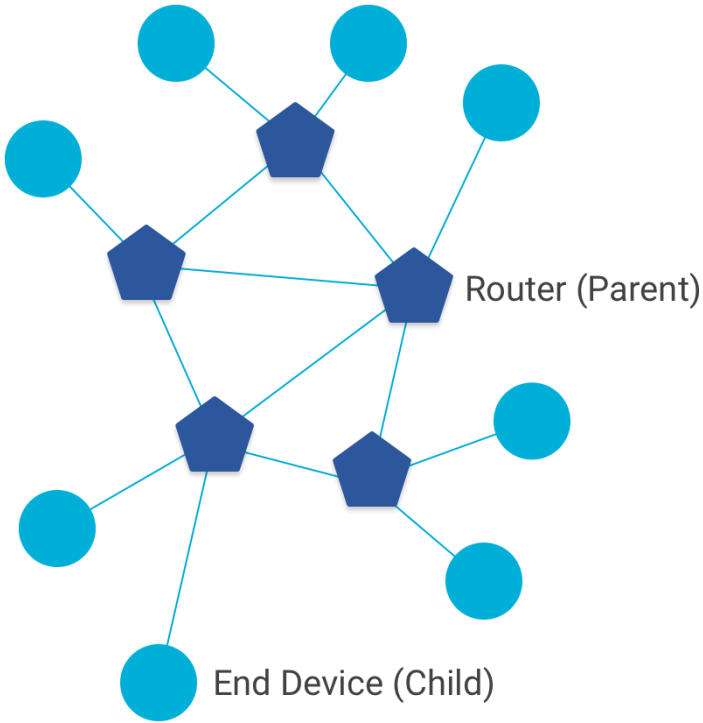
Changes to Link Layer and MAC

- Keep unslotted CSMA/CA algorithm
- Keep packet structure
- Keep Frame Types
  - Beacon
  - MAC Command
    - Beacon Request
    - Data Request
  - Data
  - Acknowledgement



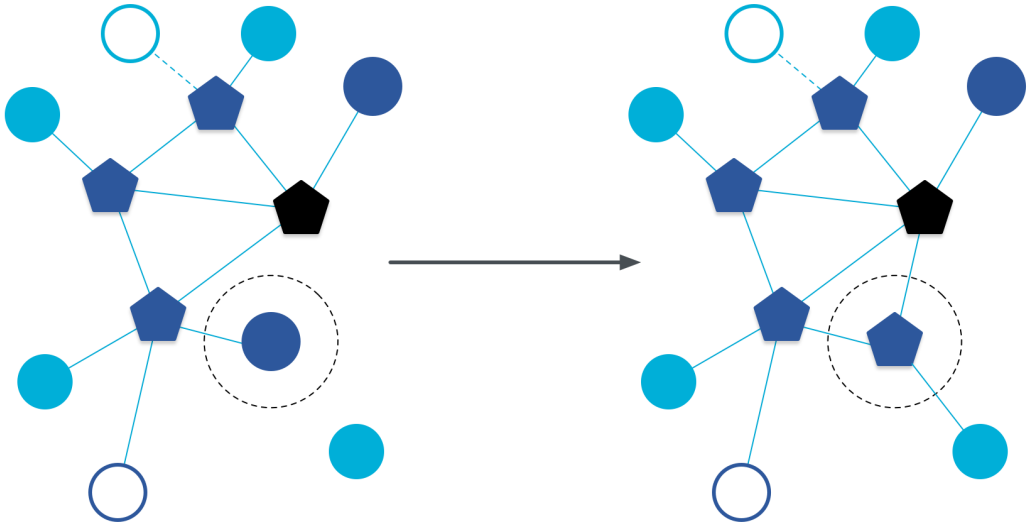
Combination of star and mesh topology

- Routers (parent)
  - Mesh communication with other routers
  - Radio always on
  - Forwards packets for network devices
  - Enables other devices to join network
  - 32 routers per network
- End devices (child)
  - Communicates with one parent (router)
  - Does not forward packets
  - Can disable transceiver to save power
    - Send packets periodically to avoid timeout
  - 511 end devices per router



Router promotion

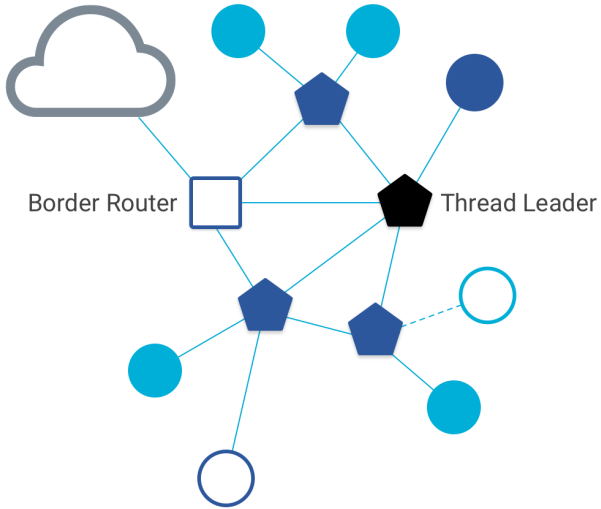
- "Router Eligible End Device"
  - A router without any children
  - Can operate as an end device with one connection (lower power)
  - Promotes to a router when a joining end device relies on it
    - If there is room for an additional router (max 32, typical 16-23)



Other special roles

- Thread leader
  - Device in charge of making decisions
    - Addresses, Joining details
  - Automatically selected from routers
    - One leader at any given time
    - Additional leader is selected if the network partitions

- Border router
  - Router that also has connectivity to another network
    - Commonly WiFi or Ethernet
  - Provides external connectivity
  - Multiple border routers may exist at once



Outline

- Thread
  - Overview
  - 6LoWPAN
  - Addressing & Routing
  - Runtime

Thread uses IPv6 for communication

- Why IP?
  - If Wireless Sensor Networks represent a future of billions of connected devices distributed throughout the physical world
  - Why shouldn't they run standard protocols wherever possible?
  - Why IPv6?
    - Generalized, Flexible, Capable
- Benefits
  - Interoperability with normal computers and networks
  - Reuse state of the art developed standards instead of remaking them
    - Security, Naming, Discovery, Services
- Costs
  - Packet overhead can be high (will fix)
  - Complexity for supporting protocols

Hui and Culler, "IP is Dead, Long Live IP for Wireless Sensor Networks". 2008

Background: IPv6

- Replacement to Internet Protocol v4
  - (Something unrelated used version number 5)
- Extended addressing for devices
  - 32-bits for IPv4 addresses -> 128-bits for IPv6 addresses
  - Example: a39b:239e:ffff:29a2:0021:20f1:aaa2:2112
- Supports multiple transmit models
  - Broadcast: one-to-all
  - Multicast: one-to-many
  - Unicast: one-to-one
- Various other improvements

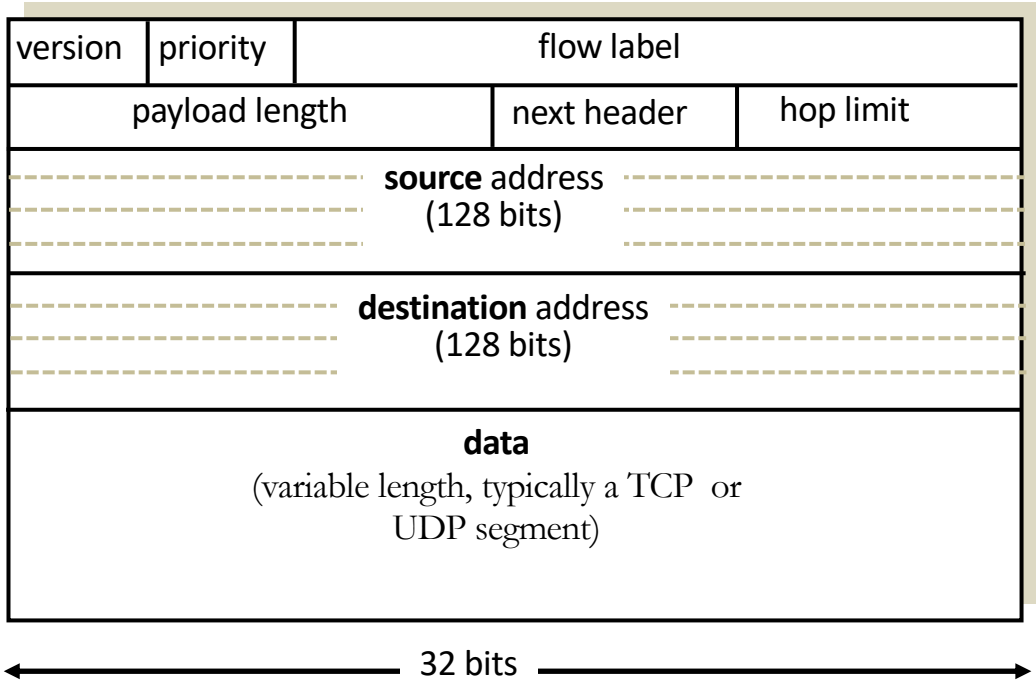
Background: IPv6 address notation rules

- Groups of zeros can be replaced with "::"
  - Can only use "::" in one place in the address
- Leading zeros in a 16-bit group can be omitted

```
0000:0000:0000:0000:0000:0000:0000:0001 → ::1
2345:1001:0023:1003:0000:0000:0000:0000 → 2345:1001:23:1003::
aecb:0222:0000:0000:0000:0000:0000:0010 → aecb:222::10
```

- Special addresses
  - Localhost - `::1` (IPv4 version is `127.0.0.1`)
  - Link-Local Network - `fe80::` (bottom 64-bits are ~device MAC address)
  - Local Network – `fc00::` and `fd00::`
  - Global Addresses – `2000::` (various methods for bottom bits; just `whois` currently)

Background: IPv6 datagram format

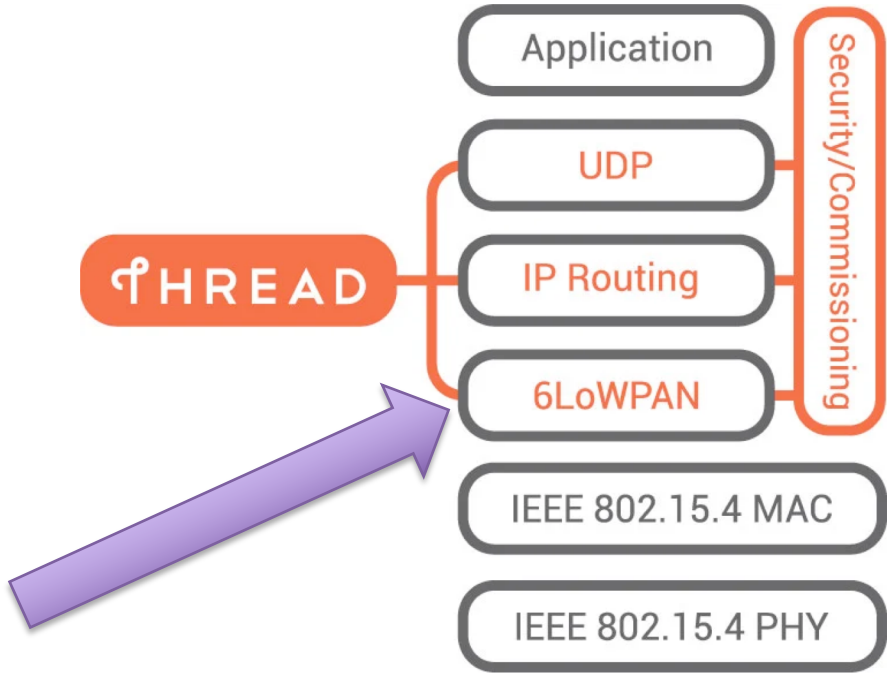


- **Priority**: like "type of service" in IPv4.
- **Flow label**: ambiguous
- **Next header**: TCP, UDP
- **Hop limit** = TTL

how much overhead?

- **40 bytes** of IPv6
- 20 more than IPv4

Thread uses *a subset* of IPv6 for communication



6LoWPAN

- Method for running IPv6 over 802.15.4 links
  - IPv6 over Low-Power Wireless Personal Area Networks
  - IETF Standard (RFC4944 + updates in RFC6282)

- 'Between' OSI Layer 2 (MAC) and Layer 3 (Network), and provides:
  - IPv6 encapsulation
  - IPv6 header compression
  - Packet fragmentation/reassembly
  - Mesh networking support ("link layer packet forwarding")

6LoWPAN IPv6 Packet Encapsulation — "Stacked Headers"

- "Pay for what you use" layering of headers:

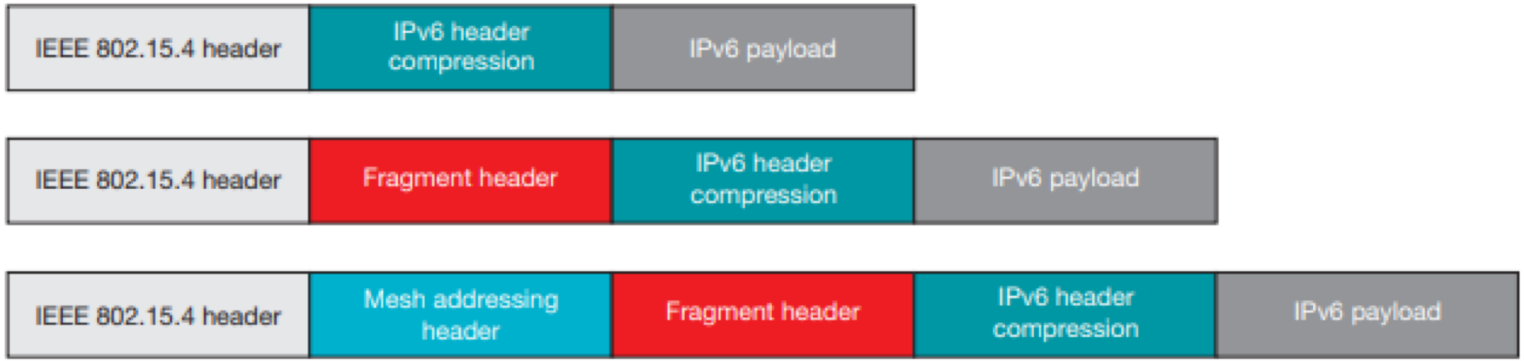


Figure 4. 6LoWPAN stacked headers

6LoWPAN header compression

- 40 bytes of IPv6 header are a lot for a 127-byte payload
  - Communication with devices in the 15.4 network should be low-overhead
  - Communication outside of the 15.4 network should still minimize overhead where possible
- While we're at it, 8 bytes of UDP header is a lot too
- Assume a bunch of common parameters to save space
  - A bunch of options are set to default values
  - Payload length can be re-determined from packet length
  - Source/Destination addresses can often be reassembled from link layer data
    - Plus information about network address assignment known by routers
  - Best case: 48 bytes IPv6+UDP → 6 bytes
- Border router "inflates" the packet before sending externally

6LoWPAN fragmentation

- Only the first packet of the fragments will hold the IPv6 header
  - Tag, offset, and size are used to reconstruct

Figure 15. First Fragment Header

Figure 16. Subsequent Fragment Header

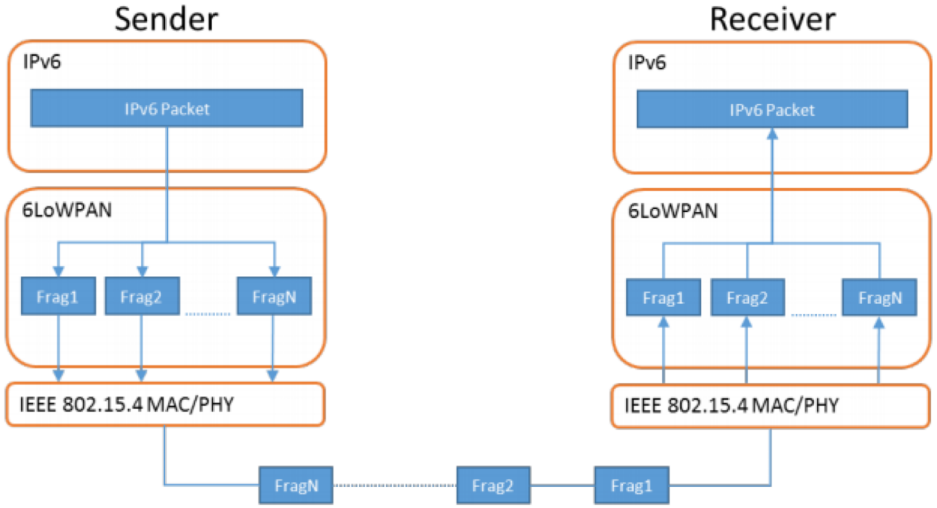


Figure 14. Fragmenting and Reassembling an IPv6 Packet

6LoWPAN mesh forwarding

- Additional header with originator and final addresses

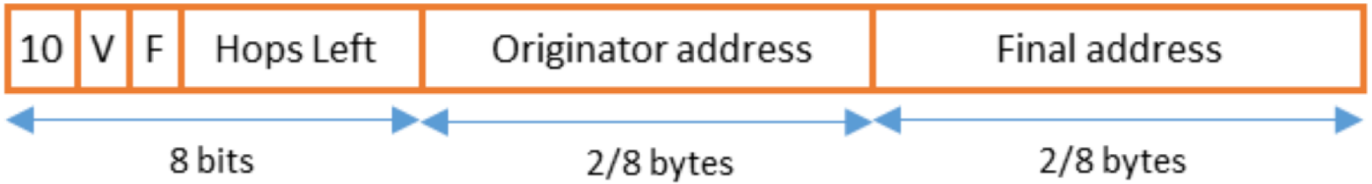
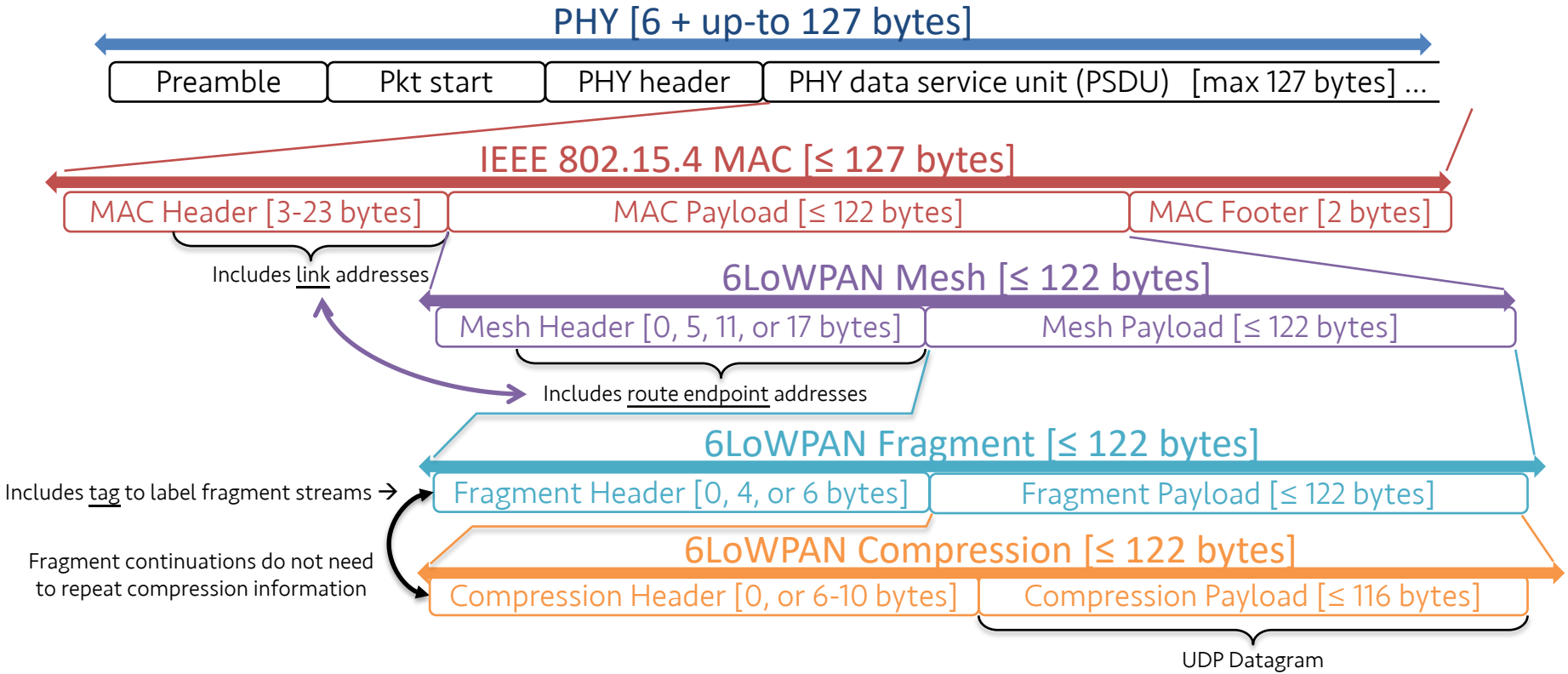


Figure 17. Mesh Header Format

Zooming out to see the whole picture

aka: Where do these headers fit again? aka: This onion has many layers



Sidebar: IPv6 over BLE

- RFC7668 defines 6LoWPAN techniques for BLE connections

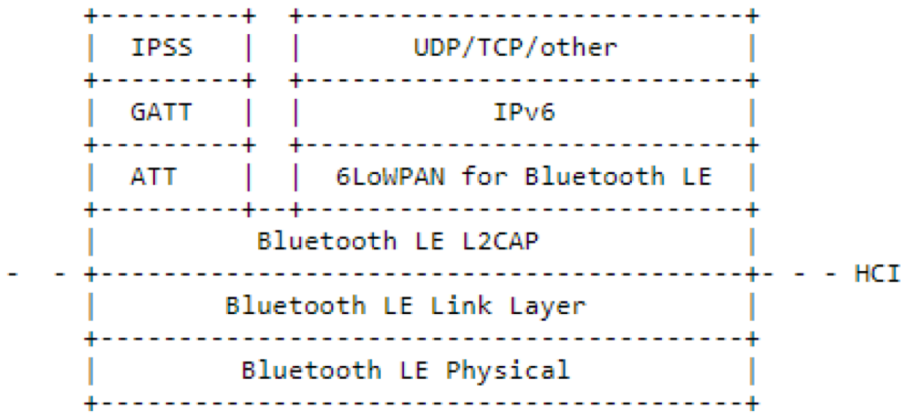


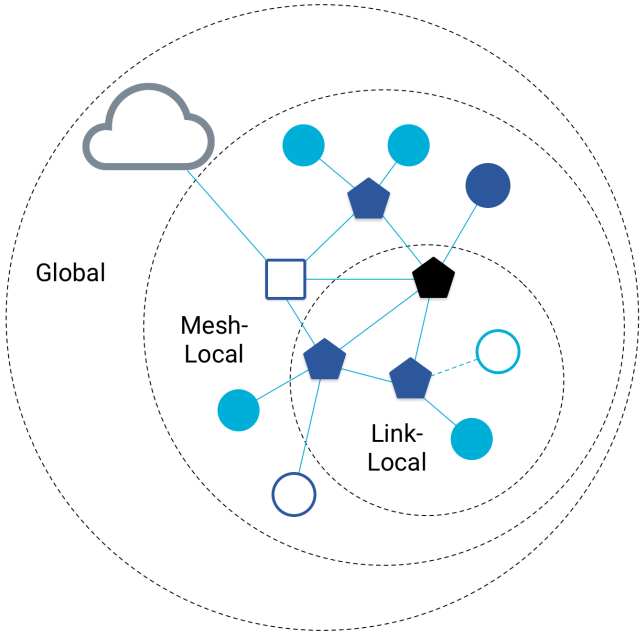
Figure 3: IPv6 and IPSS on the Bluetooth LE Stack

Outline

- Thread
  - Overview
  - 6LoWPAN
  - Addressing & Routing
  - Runtime

Benefit to IPv6: multiple address spaces per Thread device

- Each device gets an IPv6 address for each way to contact it
  - Global IP address
  - Mesh-local IP address
  - Link-local IP address
  - Topology-based IP address
  - Role-based IP address(es)



Traditional addresses in Thread

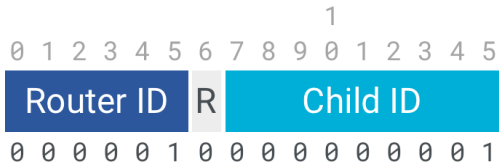
- Link-Local Addresses
  - `FE80::/16`
  - Bottommost 64-bits are EUI-64 (MAC address with 0xFFFE in the middle)
  - Permanent for a given device (no matter the network)
  - Used for low-layer interactions with neighbors (discovery, routing info)

- Mesh-Local Addresses
  - `FD00::/8` (`FD00::` and `FC00::` are for local networks)
  - Remaining bits are randomly chosen as part of joining the network
  - Permanent while connection is maintained to a network
  - Used for application-layer interactions

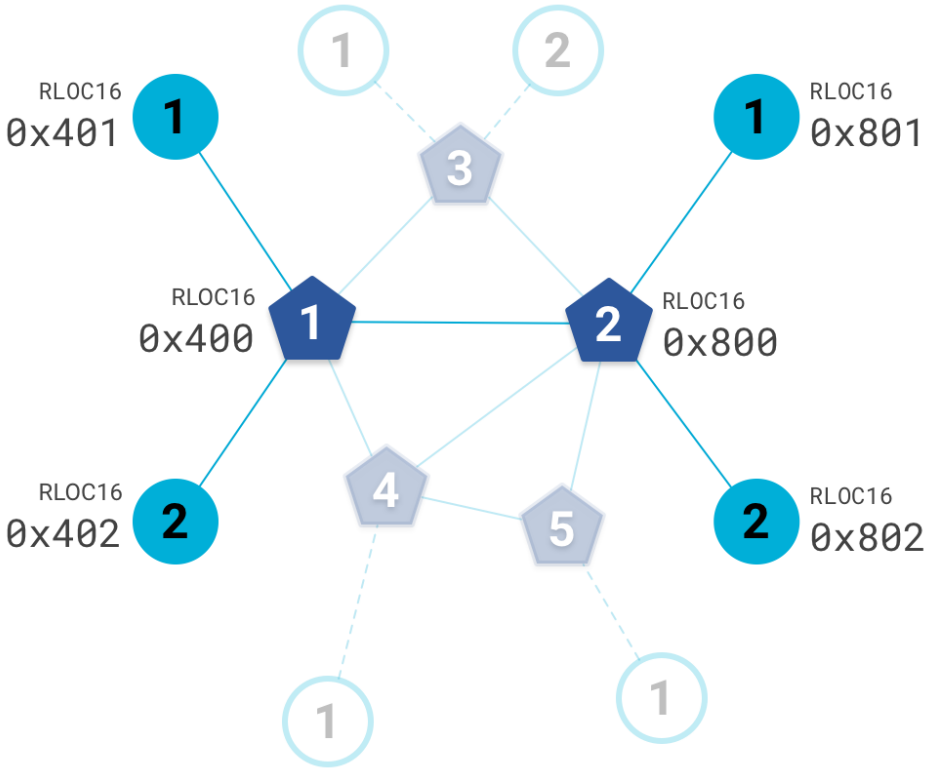
- Global Addresses
  - `2000::/3`
  - Public address for communicating with broader internet through Border Router
  - Various methods for allocation (SLAAC, DHCP, Manual)

Topology-based addresses in Thread

- `FD00::00ff:fe00:RLOC16`
  - Same top bits as mesh-local
- Routing Locator (RLOC)
  - Router ID plus Child ID



- Changes with network topology
  - Used for routing packets



Role-based addresses in Thread

- Multicast
  - `FF02::1` – link-local, all listening devices
  - `FF02::2` – link-local, all routers/router-eligible
  - `FF03::1` – mesh-local, all listening devices
  - `FF03::2` – mesh-local, all routers/router-eligible

- Anycast
  - `FD00::00FF:FE00:FCxx`
    - `00` – Thread Leader
    - `01-0F` – DHCPv6 Agent
    - `30-37` – Commissioner
    - etc.

Routing in Thread

- Only routers actually route
  - Over 'established mesh links'
- Uses a variation of RFC 2080 (aka **RIPng**)
  - Distance Vector Routing
    - How "long" is each link

Table 2. Link Quality and Link Cost

Link Margin	Link Quality	Link Cost
None (~0 dB)	0	unknown / infinite
Poor	1	4
Reasonable	2	2
Good	3	1

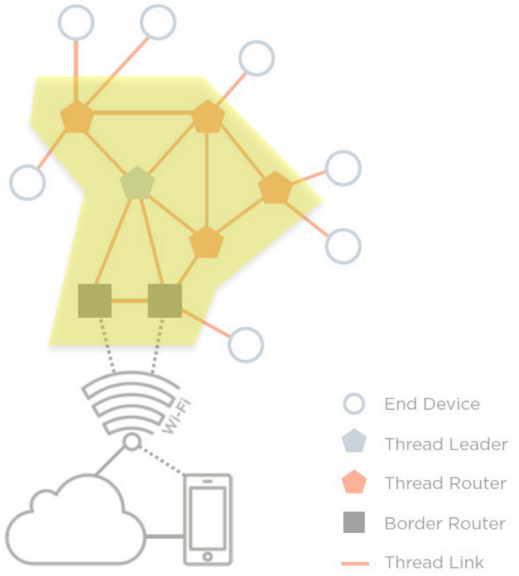
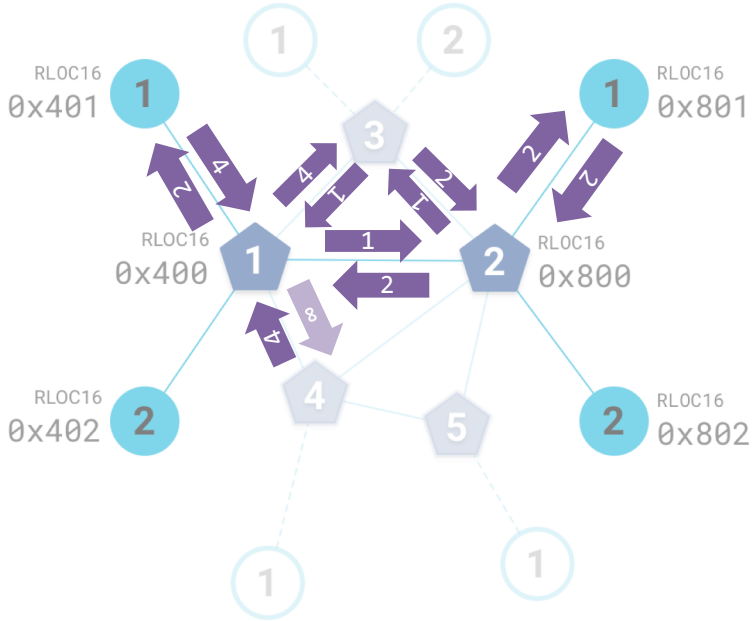


Figure 3. Basic Thread Network Topology and Devices

Distance vector routing

[Dijkstra's back, back again]

- Choose the lowest-cost path between nodes
  - Note possibly link asymmetry!



Recall, children can *only* talk to parent routers.

Child address is *set* by the parent they connect to.

So routing uses top address bits to find parent, then parent uses bottom bits to get to its child.

Routing state is flooded to all nodes

- One type of "Mesh Link Establishment" (MLE) packet
- Recall, flooding is easy!
- Implication:
  - All routers have state about *all* links
  - Thread spec says max 32 routers
  - Thus, 32 * 31 * 2 = 1024 table entries
  - Context:
    - Arudino UNO has 2 kB RAM
    - nRF528400 has 256 kB RAM [this is quite large]

Role-based addresses in Thread

- Multicast
  - `FF02::1` – link-local, all listening devices
  - `FF02::2` – link-local, all routers/router-eligible
  - `FF03::1` – mesh-local, all listening devices
  - `FF03::2` – mesh-local, all routers/router-eligible

Thread delivery is on-demand, routing is “chatty”

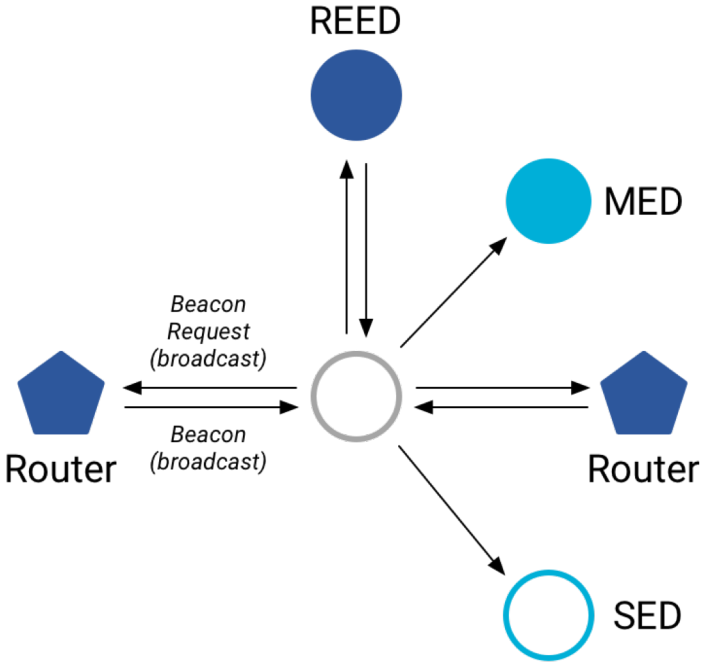
- Recall: All routers required to have radios always on
  - This means routers should always be ready to receive packets
  - Easy for edge devices (can just send to parent)
  - Easy for mesh (just send to next hop)
  - Always-on routers is a big part of what makes Thread reliable (or work at all)
- Routers continuously, proactively measure links
  - Enables source-based routing
  - Contrast to prior protocols which figured out for packets on-demand as they went through the network
  - Tradeoff: More idle overhead for more reliability [hence plugged-in routers]

Outline

- Thread
  - Overview
  - 6LoWPAN
  - Addressing & Routing
  - Runtime

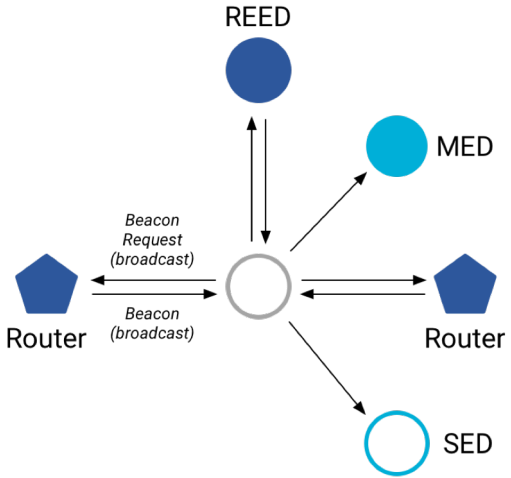
Discovering Thread networks

- Beacon request MAC command
  - Routers/Router-eligible devices respond
  - Payload contains information about network
- Thread network specification
  - PAN ID – 16-bit ID
  - XPAN ID – extended 64-bit ID
  - Network Name – human-readable
- Active scanning across channels can quickly find all existing nearby networks



Reminder of Alphabet Soup

Notice how few modes are low-power!



Router		• Route Traffic • Join requests • Security requests • Sync Thread Network Data for Leader-takeover	• Radio always on • Can downgrade to REED • Can upgrade to Leader
Leader	(Special-case Router)	• Manage REED/Router	(^same)
REED	Router-Eligible End Device	• Act as FED, but capable of upgrade to Router	• Radio always on • Can upgrade to Router
FED	Full End Device	• Can forward traffic, but cannot route • Can communicate with any node in range	• Radio always on
MED	Minimal End Device	• Only communicate to owning Parent Router • Cannot route or forward	• Radio always on
SED	Sleepy End Device	• (same as MED)	• Radio off unless comm • *Device* decides when to talk to parent router
SSED	Synchronized Sleepy End Device	• (same as MED)	• Radio off unless comm • Scheduled check-ins with parent router

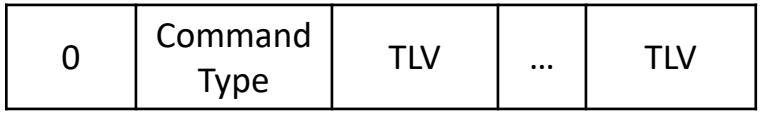
Creating a new network

- Select a channel (possibly by scanning for availability)
- Become a router
  - Elect yourself as Thread Leader
  - Respond to Beacon Requests from other devices
- Further organization occurs through Mesh-Level Establishment protocol

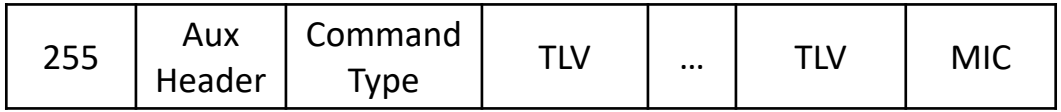
Mesh-Level Establishment

- Creating and configuring mesh links
  - Payloads placed in UDP packets within IPv6 payloads

- Commands for mesh
  - Establish link
  - Advertise link quality
  - Connect to parent



OR (secure version)

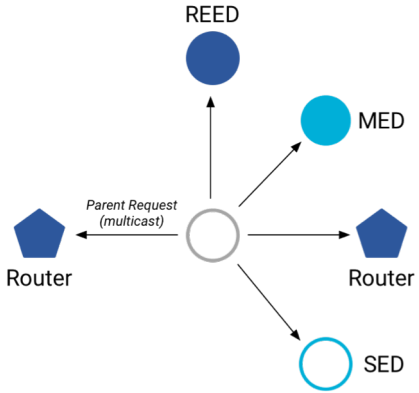


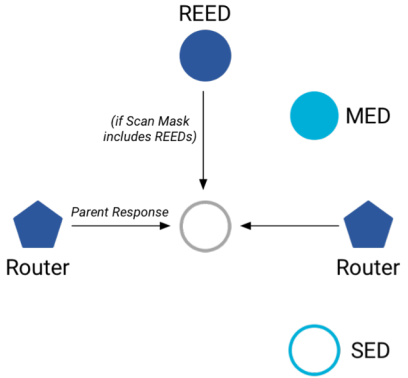
- TLVs (Type-Length-Value)
  - Various data types that may be helpful within those packets
  - Addresses, Link Quality, Routing Data, Timestamps

Joining an existing network

- All devices join as a child of some existing router

1. Send a Parent Request (to all routers/router-eligible)
   - Using the multicast, link-local address
2. Receive a Parent Response (from all routers/router-eligible separately)
   - Contains information on link quality
3. Send a Child ID Request (to router with best link)
   - Contains parameters about the new child device
4. Receive a Child ID Response (from that router)
   - Contains address configurations





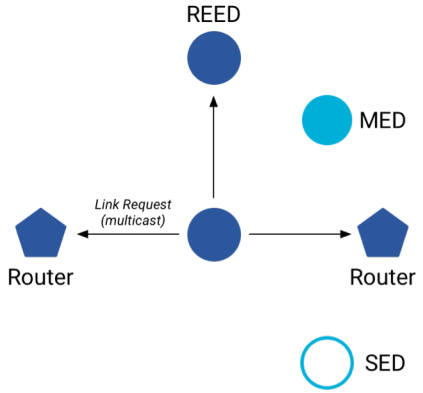
Becoming a router

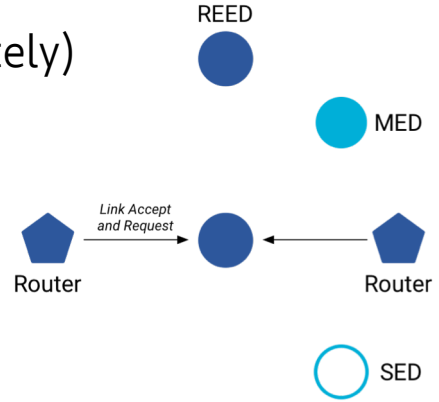
- Thread tries to maintain 16-23 routers (max 32)
  - Goals: path diversity, extend connectivity

1. Send a Link Request (to all routers/router-eligible)
   - Using the multicast, link-local address

2. Receive Link Accept and Request (from each router separately)
   - Forms bi-directional link

3. Send a Link Accept (to each router individually)





Next time: Zigbee