Wireless and Communication in the Internet of Things

Thread

Pat Pannuto, UC San Diego

ppannuto@ucsd.edu

Outline

- Mesh Refresher

- Thread
  - Overview
  - Addressing
  - Runtime Behavior

Mesh networks

- Most devices are capable of communicating with multiple neighbors

- **What are advantages of mesh?**
  - Devices can communicate over longer distances
  - Device failures less likely to collapse the entire network
- **What are disadvantages of mesh?**
  - Some nodes have to spend more energy communicating
  - Network protocol becomes more complicated to manage routing

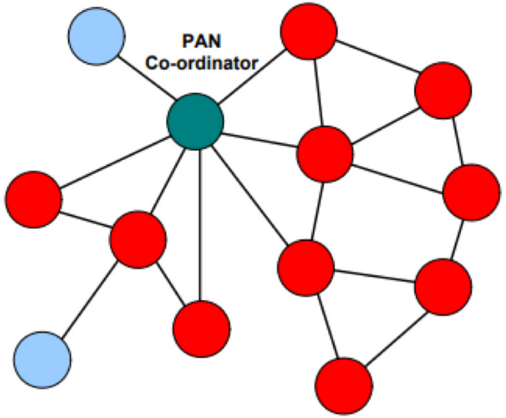


Figure 4: Mesh Topology

Quantitative intuition of ‘why bother meshing’

- Free Space Path Loss (FSPL) and the Friis transmission equation
  - Measure how RF signals travel through space
  - $FSPL = 20 * \log_{10}(\frac{4\pi R}{\lambda})$, measured in dB
  - $P_{RX} = P_{TX} * \frac{G_{TX} * G_{RX} * \lambda^2}{(4\pi R)^2 * L}$, measured in W [Friis]
  - $P_r = P_t + G_t + G_r + 20\log_{10}(\frac{\lambda}{4\pi d})$, Friis, re-written in dB

Traditionally written as transmit *d*istance

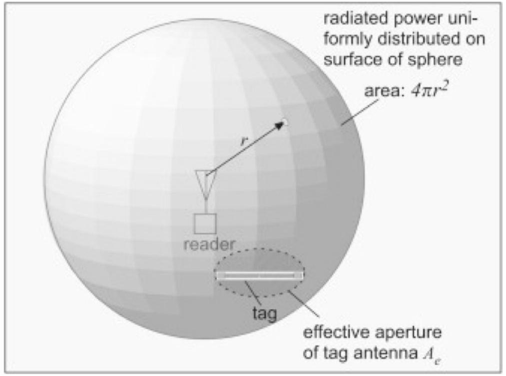


Image: https://www.sciencedirect.com/topics/computer-science/transmitting-antenna

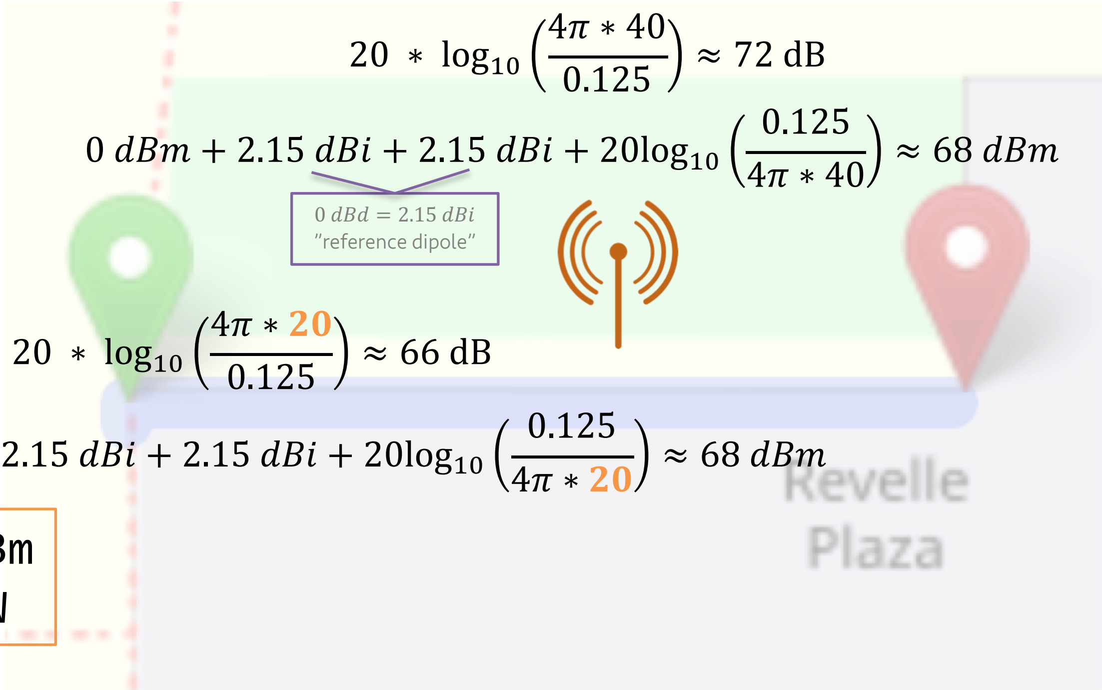
A hand-wavy quantitative analysis of ‘why bother meshing’

Revelle Plaza is ~40m wide; how to get from one side to the other?

$$\frac{3e8\ m/s}{2.4e9\ Hz} = 0.125\ m = \lambda$$

0 dBm
1 mW

-72 dBm (6.3e-8 mW)
-68 dBm (1.6e-7 mW)



$$20 * \log_{10}\left(\frac{4\pi * 40}{0.125}\right) \approx 72\ dB$$

$$0\ dBm + 2.15\ dBi + 2.15\ dBi + 20\log_{10}\left(\frac{0.125}{4\pi * 40}\right) \approx 68\ dBm$$

$0\ dBd = 2.15\ dBi$
"reference dipole"

$$20 * \log_{10}\left(\frac{4\pi * 20}{0.125}\right) \approx 66\ dB$$

$$\_\ dBm + 2.15\ dBi + 2.15\ dBi + 20\log_{10}\left(\frac{0.125}{4\pi * 20}\right) \approx 68\ dBm$$

-5.2 dBm
0.3 mW

Multi-hop mesh...
- Reduces TX power
  - But adds RX, sync cost!
- Improves aggregate network coverage
- Can improve robustness
- Reduces collision domain

Outline

- Mesh Refresher
- **Thread**
  - Overview
  - Addressing
  - Runtime

Modes of operation

- Beacon-enabled PAN
  - Slotted CSMA/CA
  - Structured communication patterns
  - Optionally with some TDMA scheduled slots
- Non-beacon-enabled PAN
  - Unslotted CSMA/CA
  - No particular structure for communication
    - Could be defined by other specifications, like Thread or Zigbee

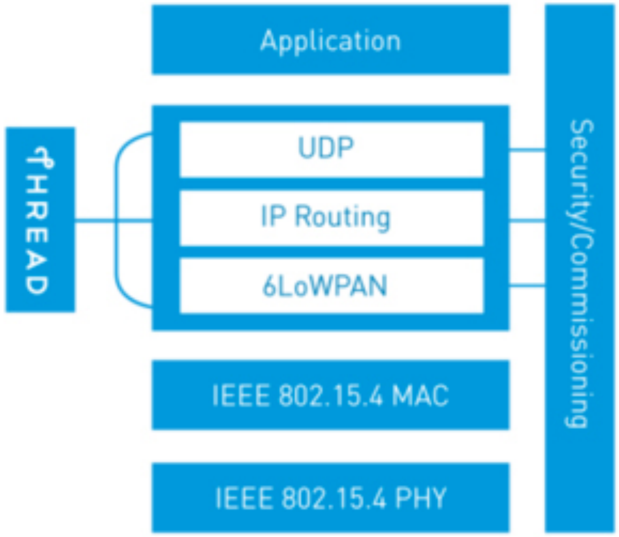
Outline

[n.b. expect to get only part way through this; probably just overview]

- Thread
  - Overview
  - Addressing
  - 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



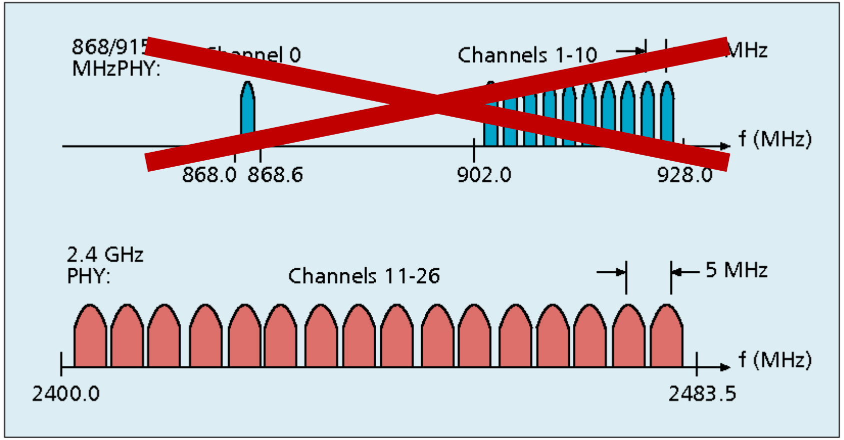
References on Thread

- 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



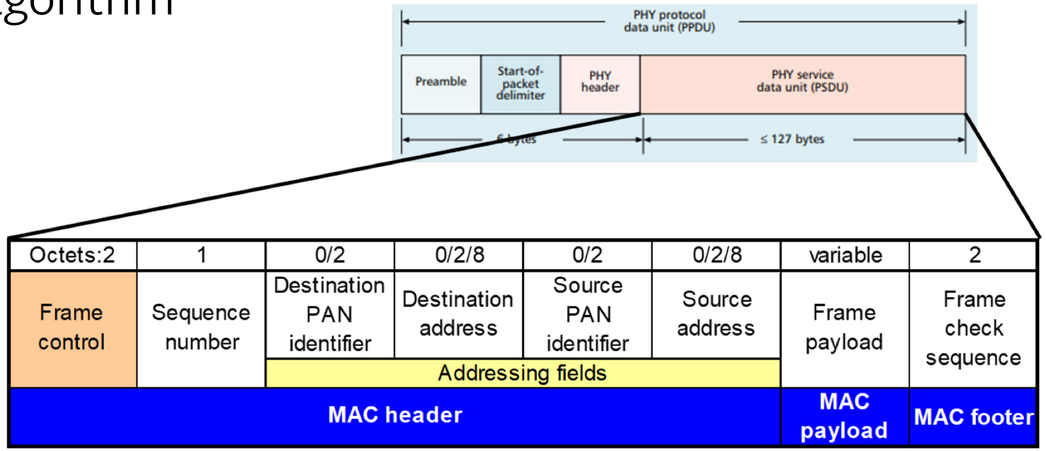
Changes to Link Layer and MAC

- 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

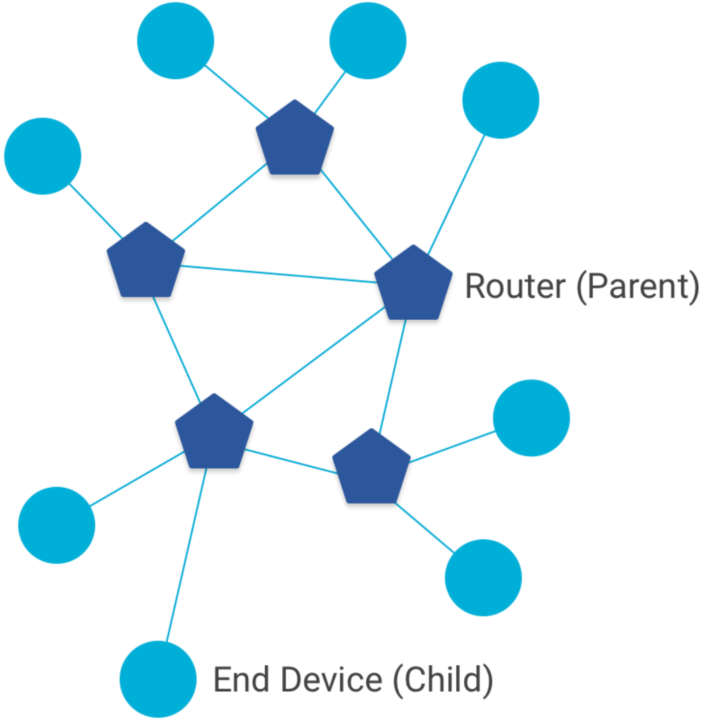
- Keep unslotted CSMA/CA algorithm
- Keep packet structure
- Keep Frame Types
  - Beacon
  - MAC Command
    - Beacon Request
    - Data Request
  - Data
  - Acknowledgement



Combination of star and mesh topology

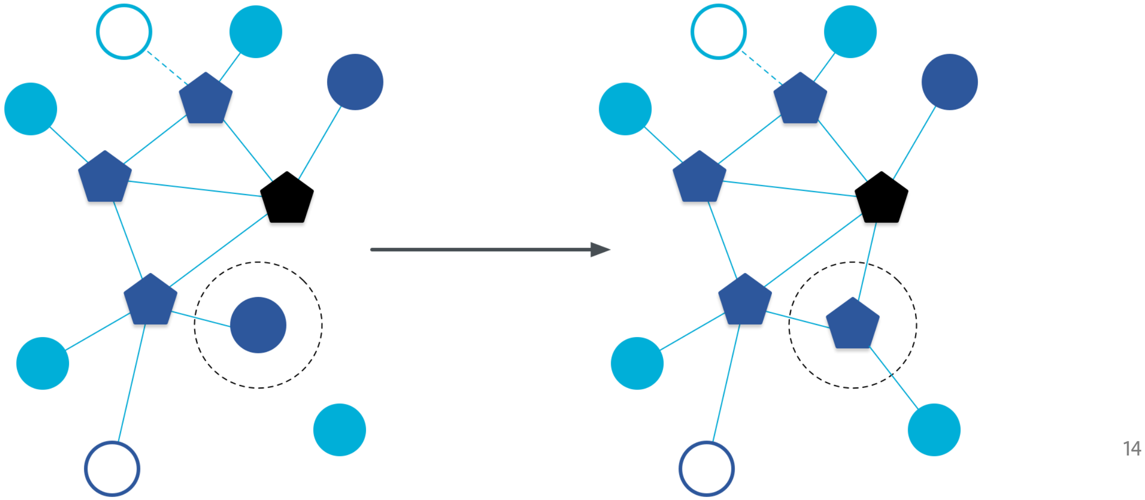
- Routers (parent)
  - Mesh communication with other routers
  - Radio always on
  - Forwards packets for network devices
  - Enables other devices to join network
  - 32 routers per network

- 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

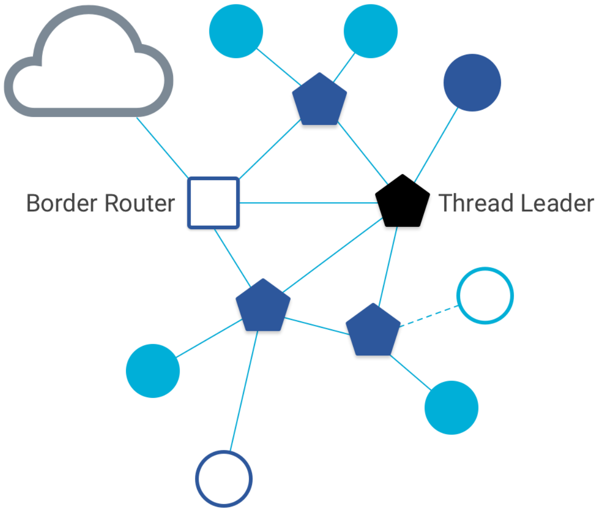
- “Router Eligible End Device”
  - A router without any children
  - Can operate as an end device with one connection (lower power)
  - Promotes to a router when a joining end device relies on it
    - If there is room for an additional router (max 32, typical 16-23)



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
[n.b. expect to get only part way through this; probably just overview]

- Thread
  - Overview
  - Addressing
  - 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

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

Thread uses IPv6 for communication

- Benefits
  -
  -
    -
- Costs
  -
  -

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

Thread uses IPv6 for communication

- 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

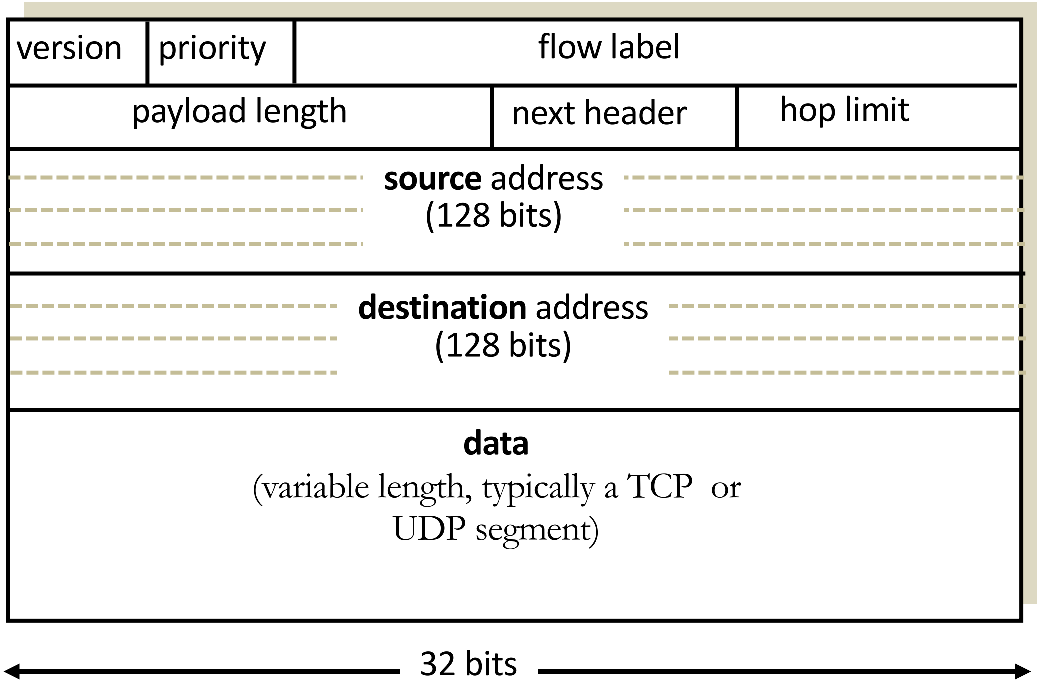
Background: IPv6 address notation rules

- 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

6LoWPAN

- Method for running IPv6 over 802.15.4 links
  - IPv6 over Low-Power Wireless Personal Area Networks
  - IETF Standard (RFC4944 + updates in RFC6282)
- Directly out of the research world (Jonathan Hui + David Culler)
  - Research Paper: IP is Dead, Long Live IP for Wireless Sensor Networks
  - Thesis of work: sensor networks can and should use IPv6
- Important goals
  - Compress IPv6 headers
  - Handle fragmentation of packets
  - Enable sending packets through mesh

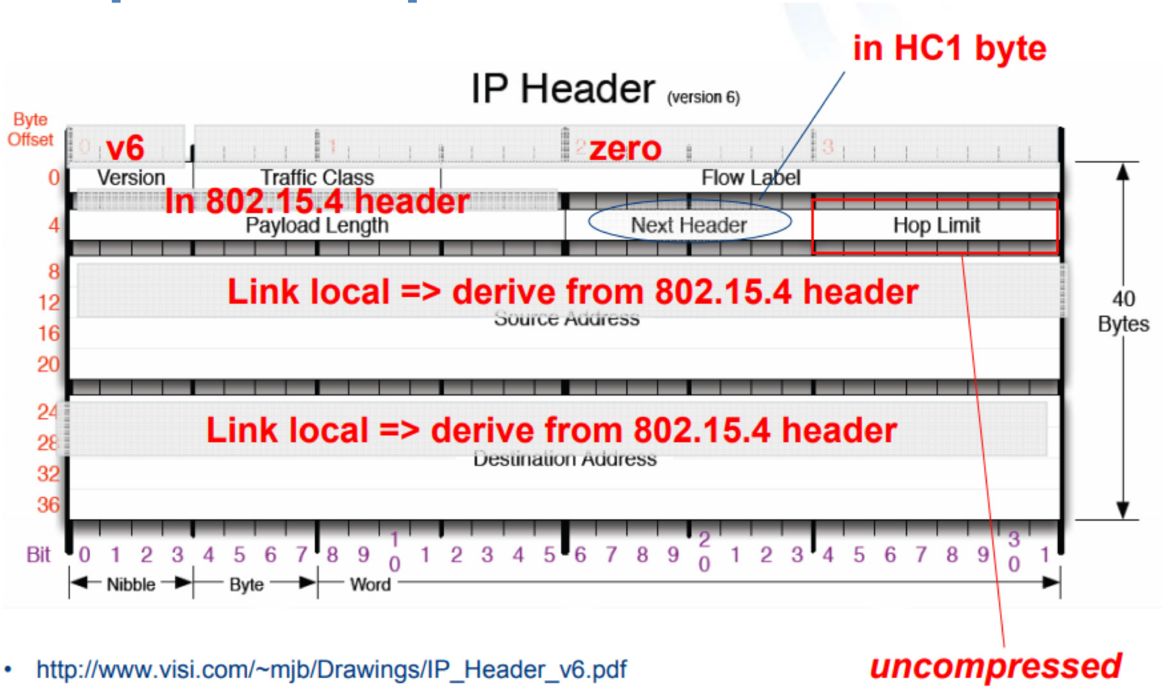
6LoWPAN header compression

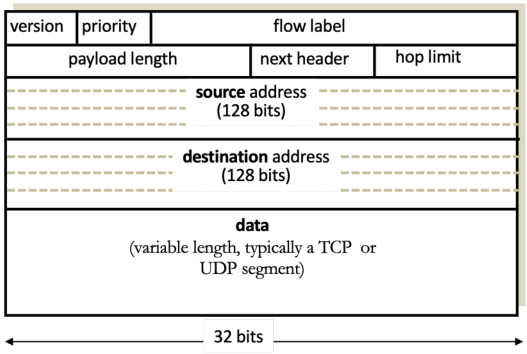
- 40 bytes of IPv6 header are a lot for a 127-byte payload
- Most important goals
  - Communication with devices in the 15.4 network should be low-overhead
  - Communication outside of the 15.4 network should still minimize overhead where possible

- 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

- Border router "inflates" the packet before sending externally

Example of compression





- http://www.visi.com/~mjb/Drawings/IP_Header_v6.pdf

- Note: Thread actually uses IPHC (not HC1) from rfc6282

6LoWPAN fragmentation

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

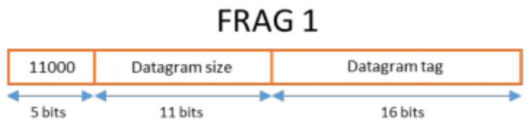


Figure 15. First Fragment Header

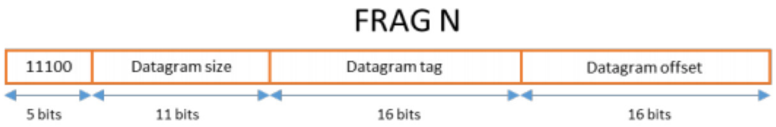


Figure 16. Subsequent Fragment Header

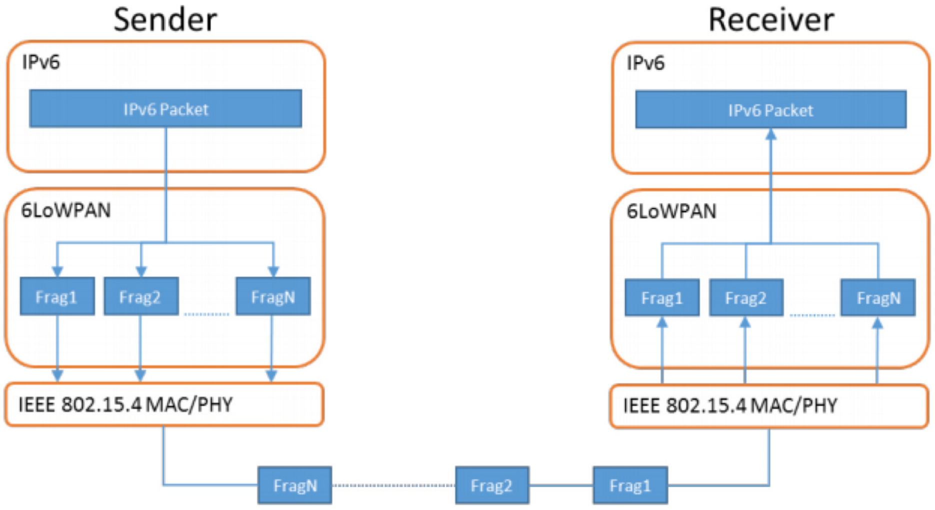


Figure 14. Fragmenting and Reassembling an IPv6 Packet

6LoWPAN mesh forwarding

- Additional header with originator and final addresses

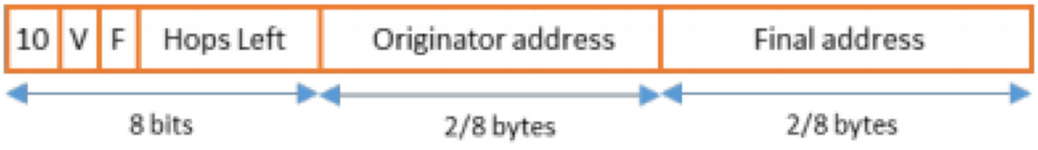


Figure 17. Mesh Header Format

- Which of these headers are used depends on the packet

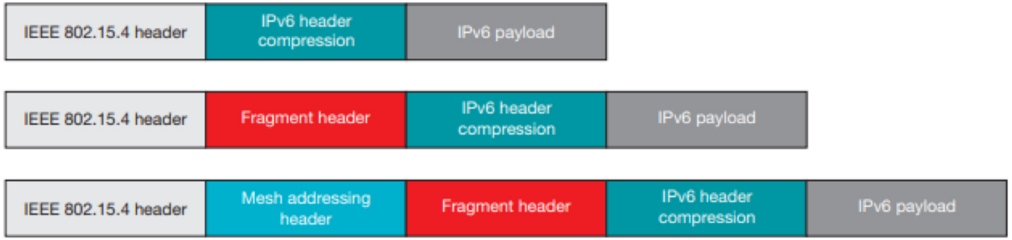


Figure 4. 6LoWPAN stacked headers

Sidebar: IPv6 over BLE

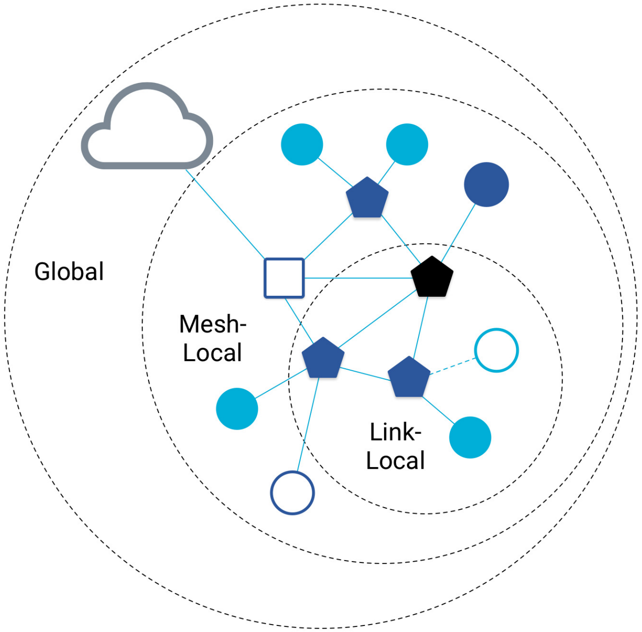
- RFC7668 defines 6LoWPAN techniques for BLE connections

```
     +---------+  +----------------------------+
     |  IPSS   |  |       UDP/TCP/other        |
     +---------+  +----------------------------+
     |  GATT   |  |            IPv6            |
     +---------+  +----------------------------+
     |  ATT    |  |  6LoWPAN for Bluetooth LE  |
     +---------+--+----------------------------+
     |          Bluetooth LE L2CAP             |
-  - +-----------------------------------------+- - - HCI
     |        Bluetooth LE Link Layer          |
     +-----------------------------------------+
     |         Bluetooth LE Physical           |
     +-----------------------------------------+

       Figure 3: IPv6 and IPSS on the Bluetooth LE Stack
```

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

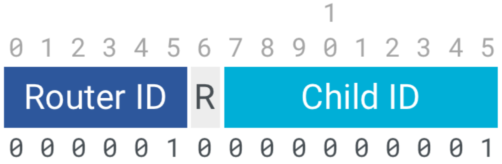
- Link-Local Addresses
  - `FE80::/16`
  - Bottommost 64-bits are EUI-64 (MAC address with 0xFFFE in the middle)
  - Permanent for a given device (no matter the network)
  - Used for low-layer interactions with neighbors (discovery, routing info)

- 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

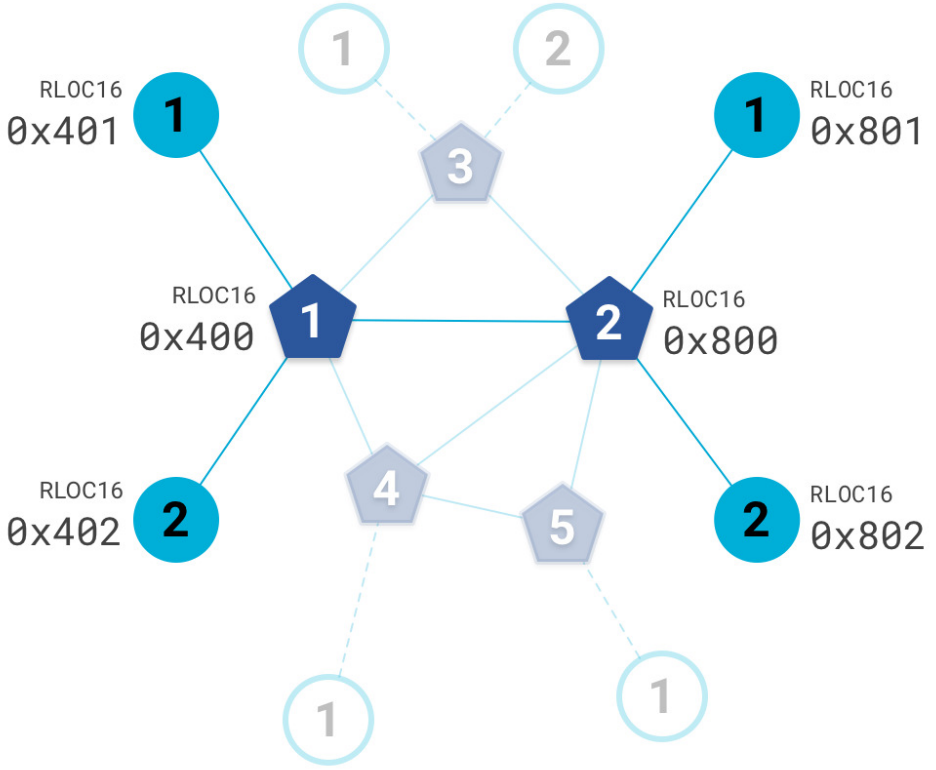
- Global Addresses
  - `2000::/3`
  - Public address for communicating with broader internet through Border Router
  - Various methods for allocation (SLAAC, DHCP, Manual)

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.

Break + Open Question

- Why use Thread instead of original 802.15.4?

Break + Open Question

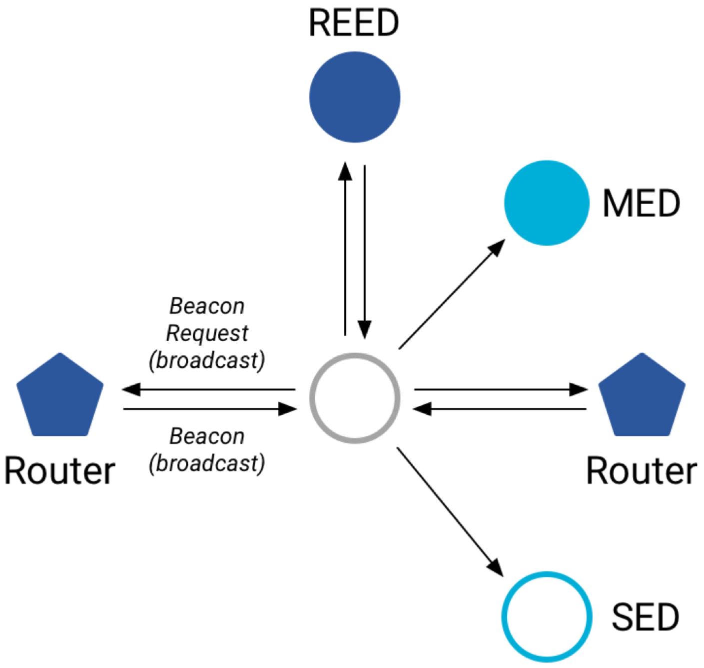
- Why use Thread instead of original 802.15.4?
  - Full specification of upper layers
    - Clarifies how data is transmitted between devices on a network
    - Cleans up a lot of things otherwise left implementation-dependent
  - Interaction with the world *outside* of the sensor network!
    - Gateway can be a dumb forwarder of packets
    - Devices can directly talk to NTP servers or POST data to a website!

Outline
[n.b. expect to get only part way through this; probably just overview]

- Thread
  - Overview
  - Addressing
  - 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



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

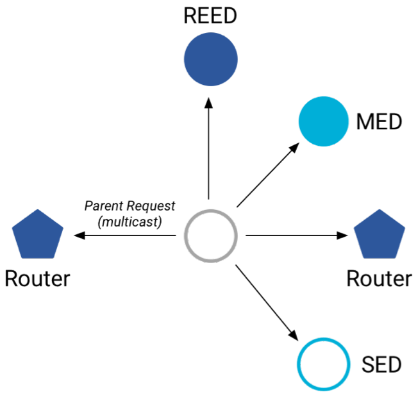
0	Command Type	TLV	…	TLV

OR (secure version)

255	Aux Header	Command Type	TLV	…	TLV	MIC

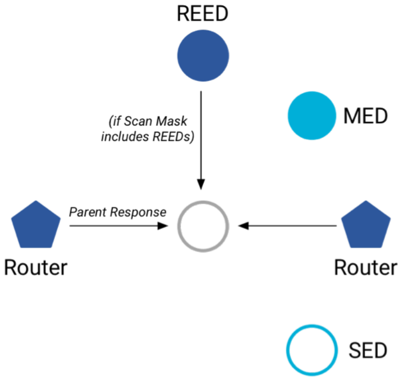
- TLVs (Type-Length-Value)
  - Various data types that may be helpful within those packets
  - Addresses, Link Quality, Routing Data, Timestamps

Joining an existing network



- All devices join as a child of some existing router

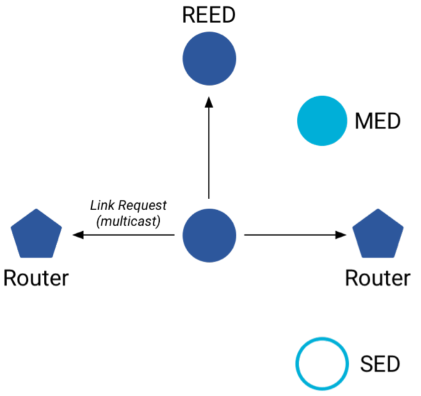
1. Send a Parent Request (to all routers/router-eligible)
   – Using the multicast, link-local address
2. Receive a Parent Response (from all routers/router-eligible separately)
   – Contains information on link quality
3. Send a Child ID Request (to router with best link)
   – Contains parameters about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

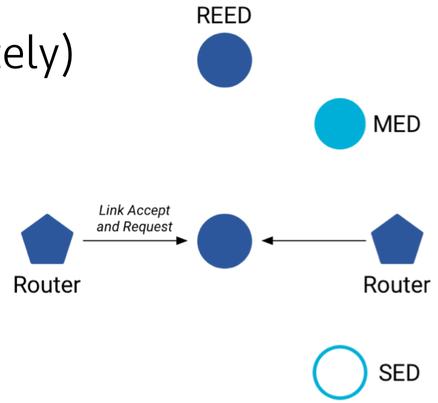


Becoming a router

- 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: More Thread, Maybe Zigbee