Lecture 08
Bonus Thread

CS397/497 – Wireless Protocols for IoT

Branden Ghena – Winter 2021

With some advice from Neal Jackson (UC Berkeley)

Today’s Goals

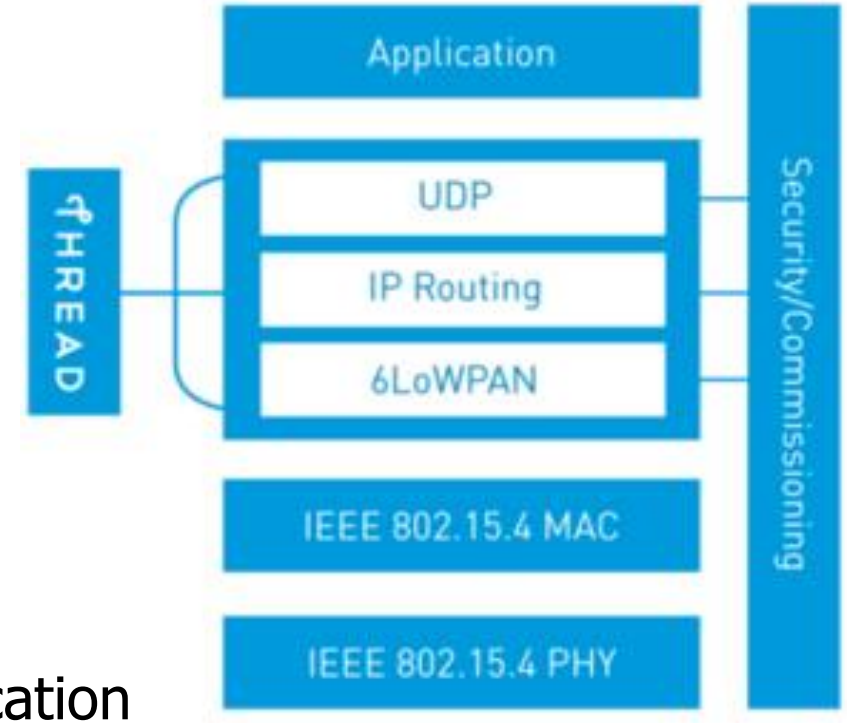
- Wallow in disappointment that the lab isn’t working
- Understand addressing in Thread networks
- Describe runtime behaviors like network joining
- Discuss uses of IP in sensor networks

Outline

- **Thread Addressing**
- Runtime Behavior
- Using IP

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

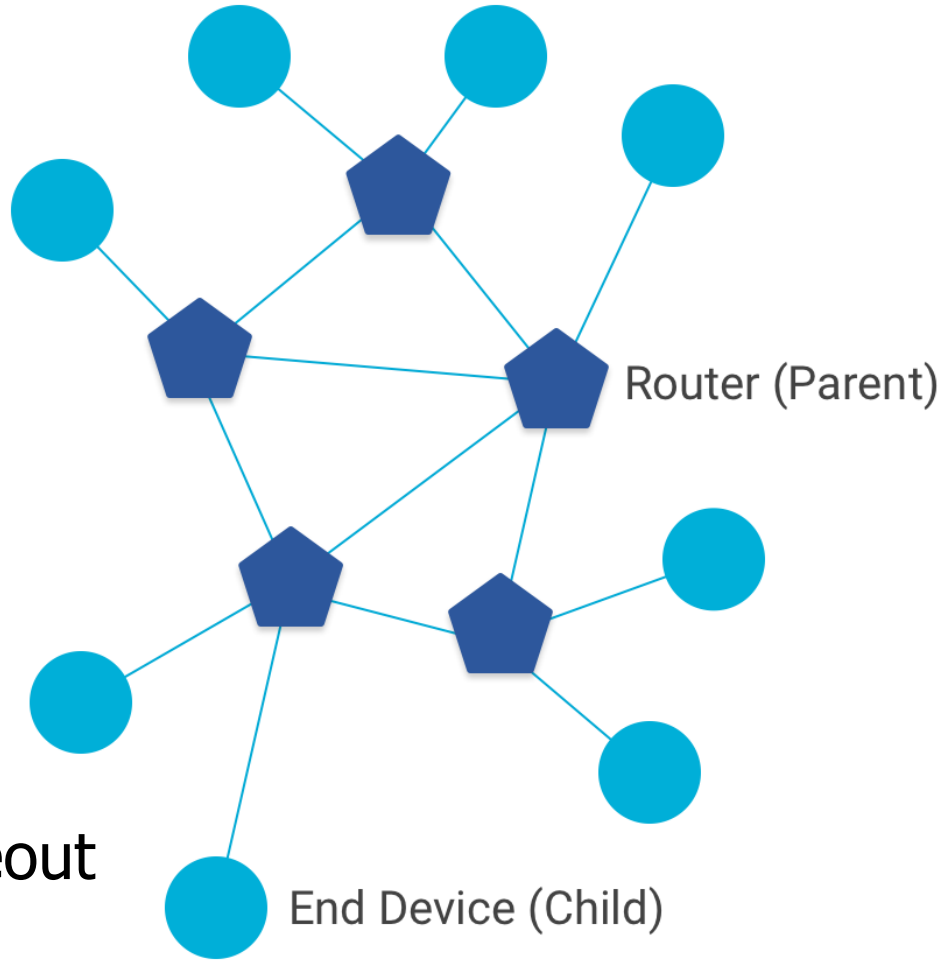


- Goals
  - Simplicity – easy to install and operate
  - Efficiency – years of operation on batteries
  - Scalability – hundreds of devices in a network
  - Security – authenticated and encrypted communication
  - Reliability – mesh networking without single point of failure

- Industry-focused, but based in academic research

Combination of star and mesh topology

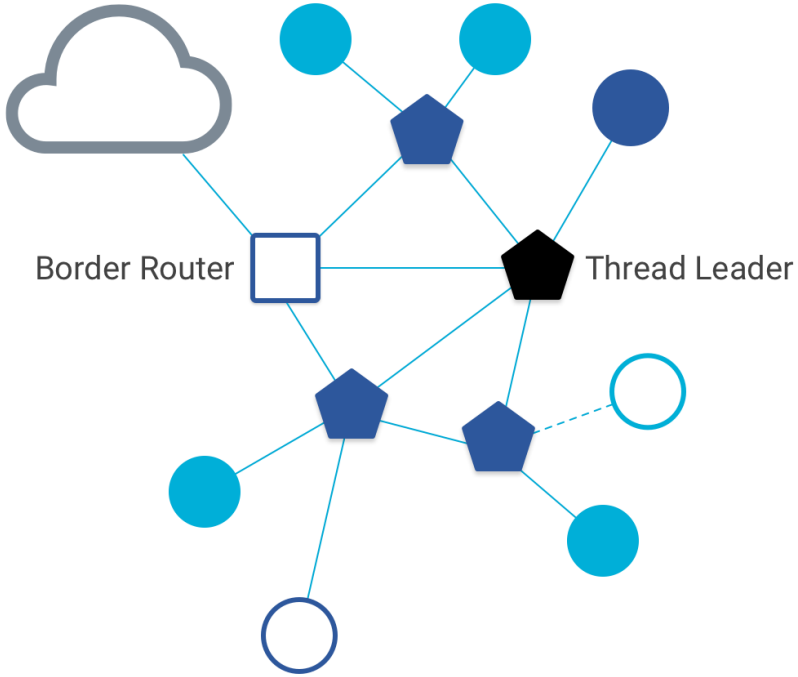
- Routers (parent)
  - Mesh communication with other routers
  - Radio always on
  - Forwards packets for network devices
  - Enables other devices to join network
  - 32 routers per network
- End devices (child)
  - Communicates with one parent (router)
  - Does not forward packets
  - Can disable transceiver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



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



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

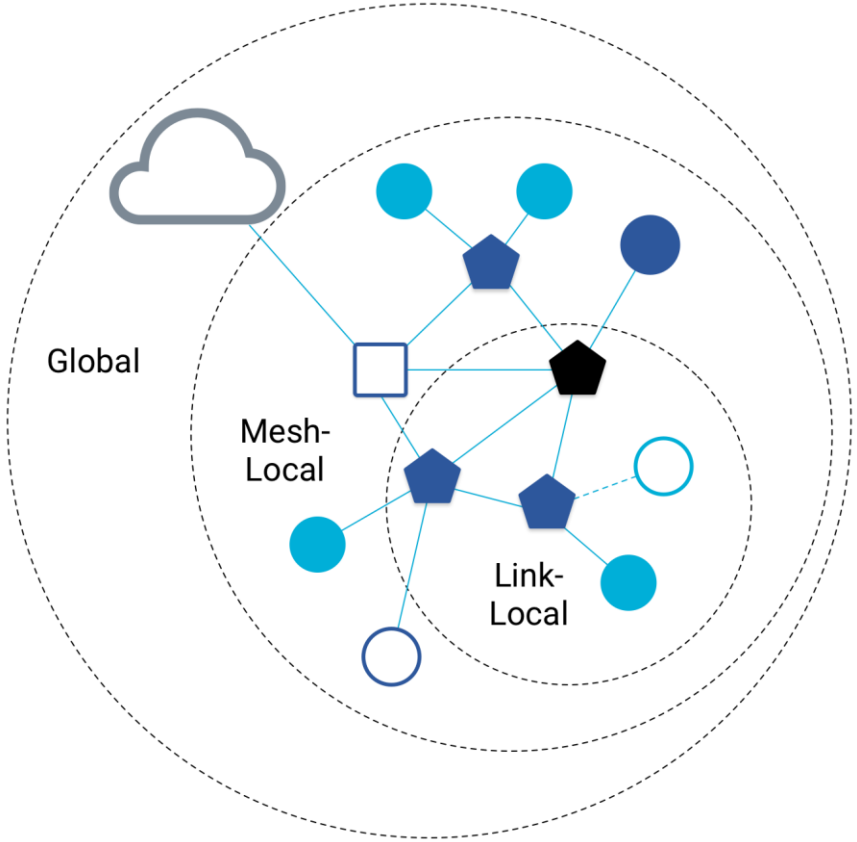
  2345:1001:0023:1003:0000:0000:0000:0000 → 2345:1001:23:1003::

  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 allocating bottom bits)

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

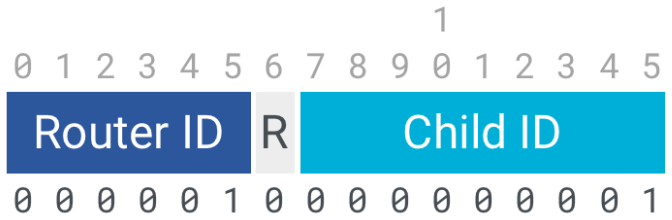
- Link-Local Addresses
  - FE80::/16
  - Bottommost 64-bits are EUI-64 (MAC address with 0xFFFE in the middle)
  - Permanent for a given device (no matter the network)
  - Used for low-layer interactions with neighbors (discovery, routing info)

- 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

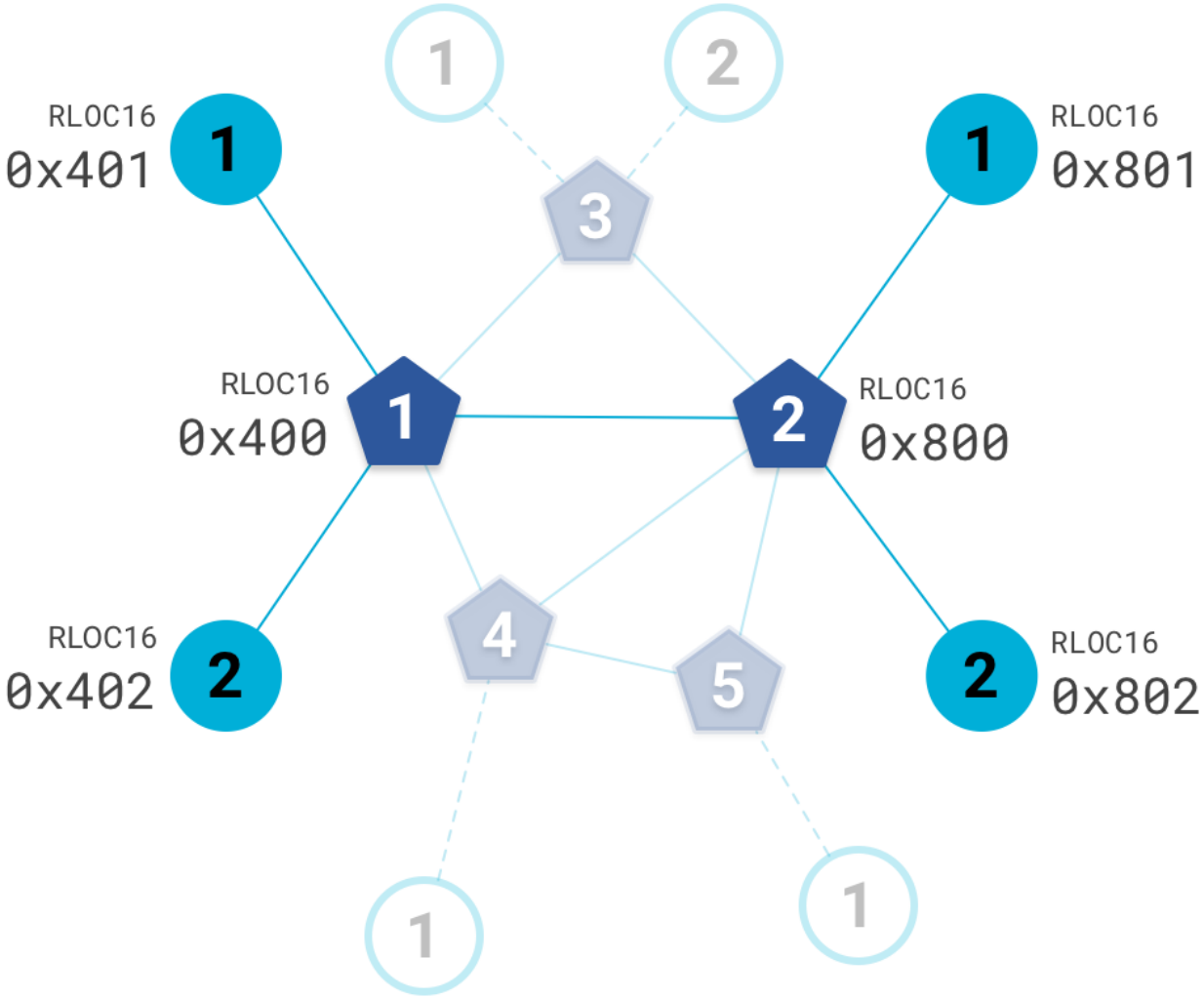
- Global Addresses
  - 2000::/3
  - Public address for communicating with broader internet through Border Router
  - Various methods for allocation (SLAAC, DHCP, Manual)

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

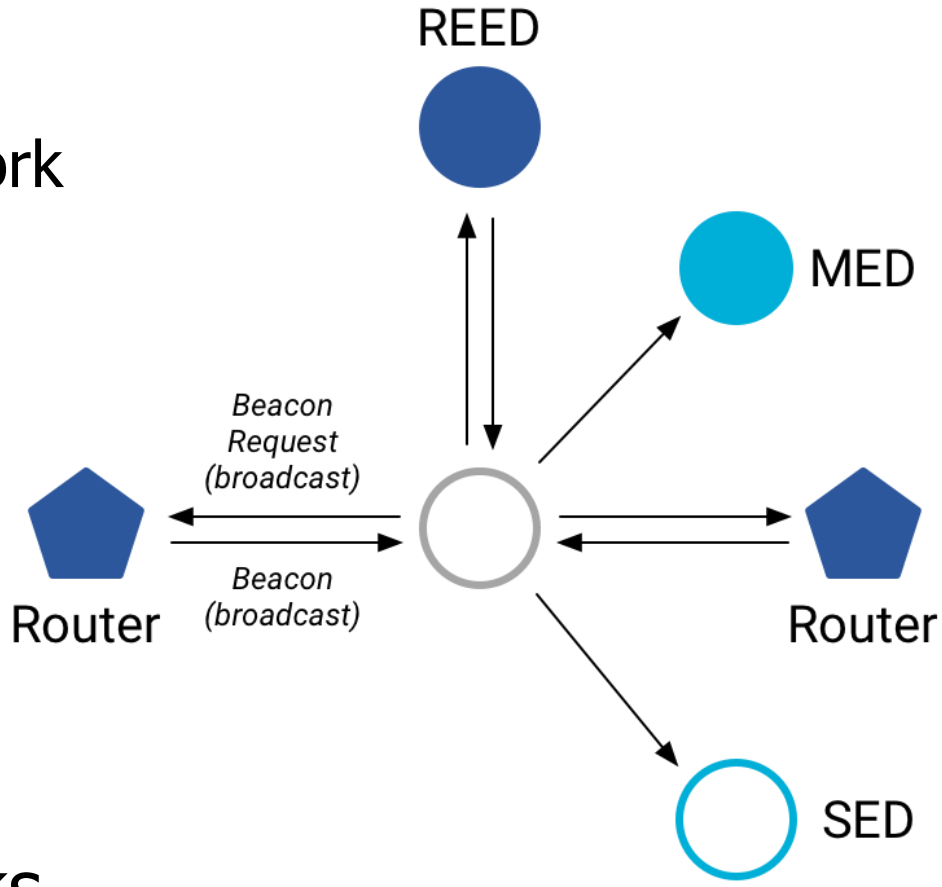
- Anycast
  - FD00::00FF:FE00:FC**xx**
    - 00 – Thread Leader
    - 01-0F – DHCPv6 Agent
    - 30-37 – Commissioner
    - etc.

Outline

- Thread Addressing
- **Runtime Behavior**
- Using IP

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

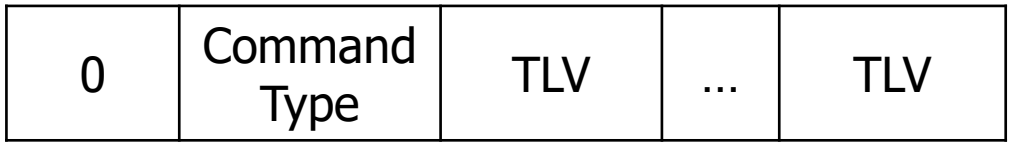
- Select a channel (possibly by scanning for availability)

- Become a router
  - Elect yourself as Thread Leader
  - Respond to Beacon Requests from other devices

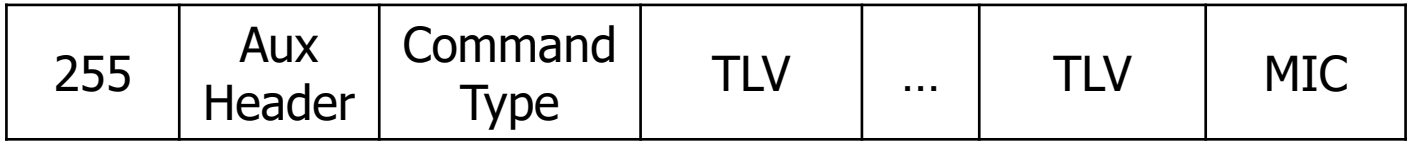
- Further organization occurs through Mesh-Level Establishment protocol

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



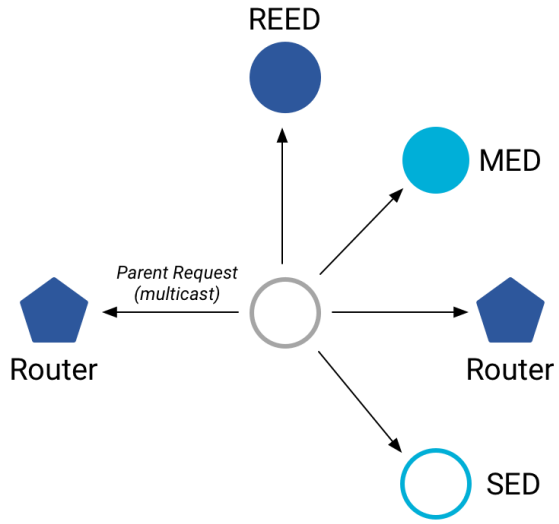
OR (secure version)



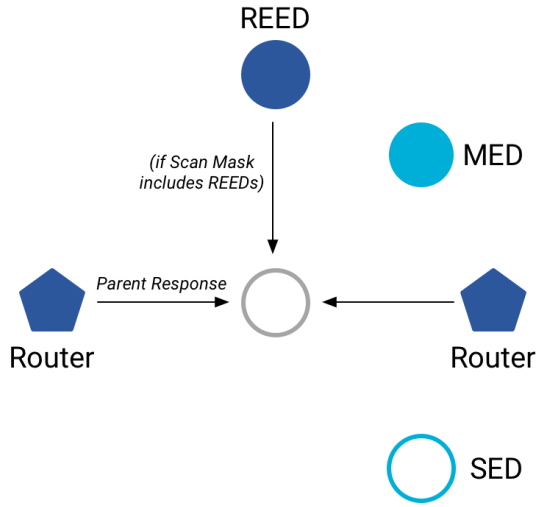
- TLVs (Type-Length-Value)
  - Various data types that may be helpful within those packets
  - Addresses, Link Quality, Routing Data, Timestamps

Joining an existing network

- All devices join as a child of some existing router

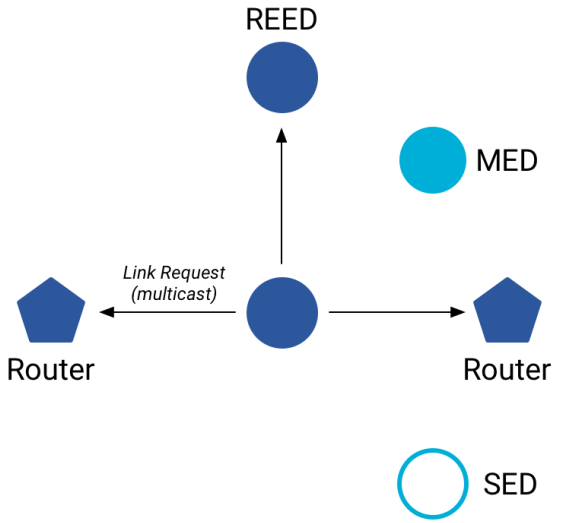


1. Send a Parent Request (to all routers/router-eligible)
   - Using the multicast, link-local address
2. Receive a Parent Response (from all routers/router-eligible separately)
   - Contains information on link quality
3. Send a Child ID Request (to router with best link)
   - Contains parameters about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



Becoming a router

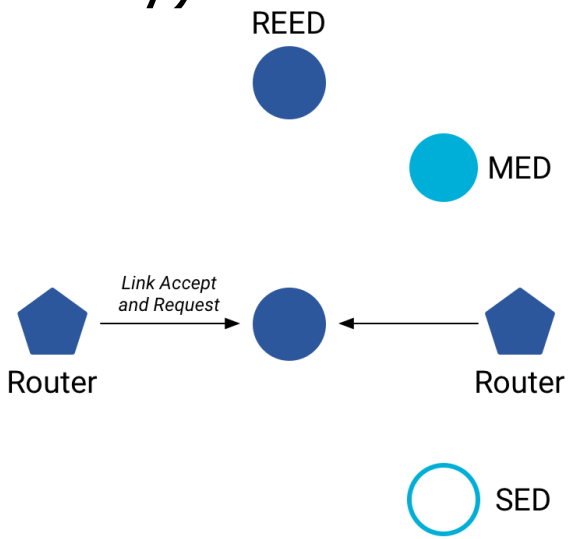
- Thread tries to maintain 16-23 routers (max 32)
  - Goals: path diversity, extend connectivity



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)



Outline

- Thread Addressing
- Runtime Behavior
- **Using IP**

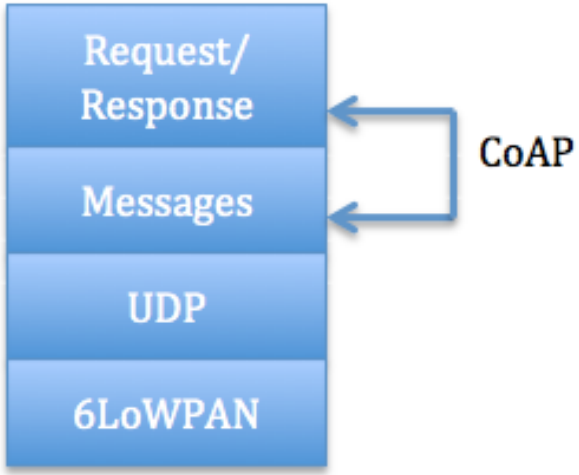
Communicating with IP

- Any communication that layers on top of IP is now possible
  - If there is a library to support it

- Common choices
  - UDP
    - DNS – translate hostnames into IP addresses
    - SNTP – get real-world time, accuracy better than 1 second
    - CoAP – send and receive data

Constrained Application Protocol - CoAP

- HTTP, but over UDP targeting less-capable devices
  - Same REST architecture
  - Adds capability for automatic retransmissions



- CoAP Requests
  - Have a type: GET, POST, PUT, DELETE
  - Have a URL: /file/etc
  - Have data up to 65 KB

Sensor networks don’t use TCP (yet?)

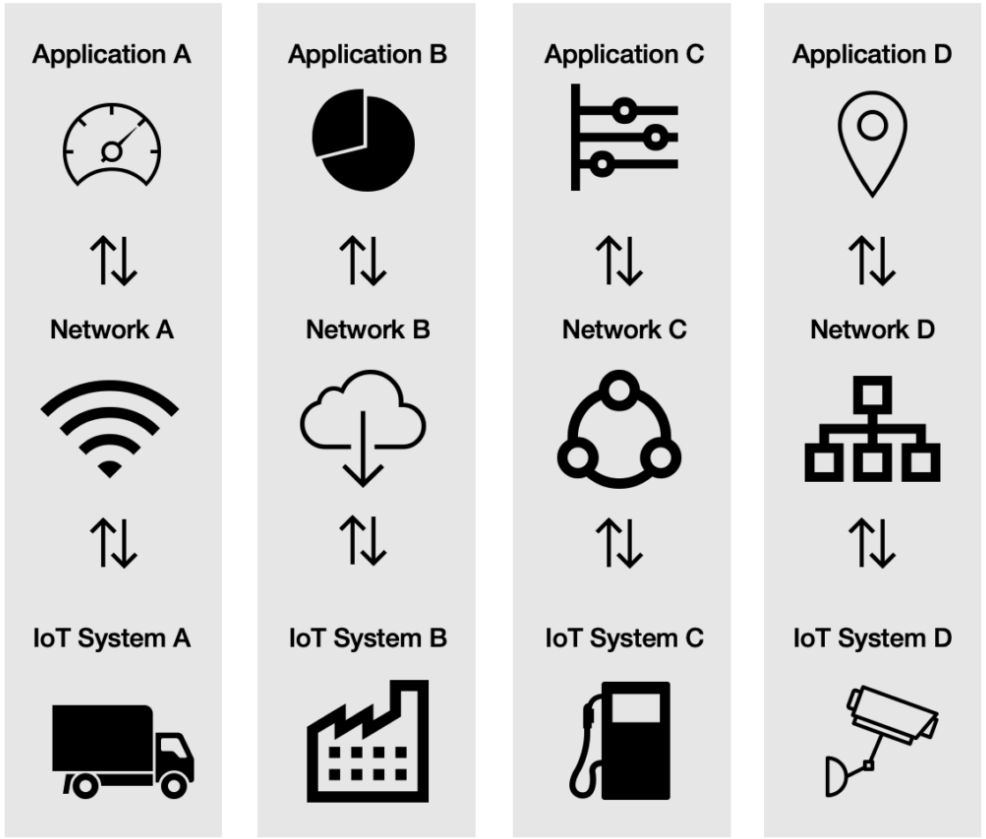
- Uncommon choice: TCP
  - Concerns: Too large, too slow, poorly suited to lossy networks
  - Also concerning: We’re just replicating TCP poorly

  - Work in progress:
    - Sam Kumar, Michael Anderson, Hyung-Sin Kim, David Culler. “Performant TCP for Low-Power Wireless Networks”. 2020.

    - The debate is still very much open

A problem: the siloed internet of things

- Problem: companies are more interested in selling you the whole stack
  - Which then makes it harder for devices to be interoperable
- This is not Thread or IP-specific, but a problem all IoT devices are facing
- IP question:
  - What IP address do you send data to?
  - Manufacturer's server is an obvious choice



Outline

- Thread Addressing
- Runtime Behavior
- Using IP