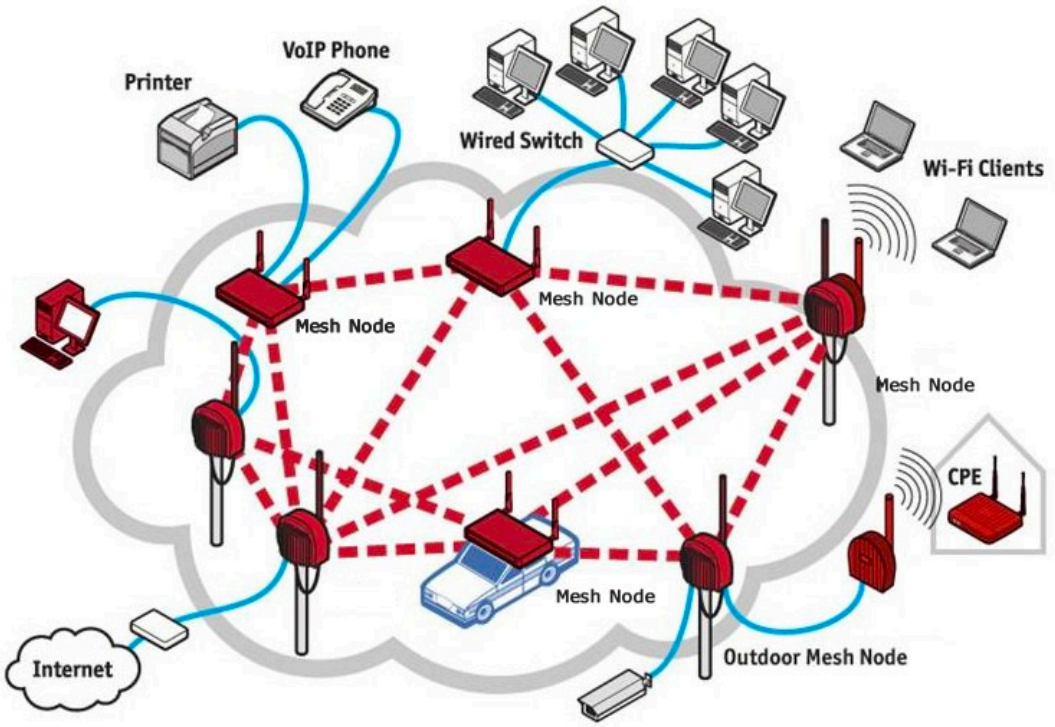
Mesh Networks



Advantages of Mesh Networking

- Self-forming
  - The wireless mesh network forms automatically once the mesh nodes have been configured and activated.
- Fault tolerance / Self-healing
  - If redundant routes exist in the network, information flow is not interrupted in the rest of the network when one node fails. The network will dynamically reroute the information via the next available route.
  - Once restored, a node rejoins the mesh network seamlessly.
- Low cost of infrastructure
  - Mesh nodes can be built from low cost, common-off-the-shelf equipment.
- Community ownership
  - Ownership of the network is shared; the burden of network support does not rest with a single person.
  - E.g., https://www.nycmesh.net/blog/how/

Mesh networks for IoT

- Mesh networking is a promising approach to organizing device-to-device communication within Internet of Things (IoT) platforms, too
- With mesh, IoT networks can work faster and more efficiently without requiring expensive hardware or consuming too much power.
- Mesh networking is a key technology-agnostic enabler for IoT in the short range space. Well-known technologies such as Wi-Fi and ZigBee have already standardized mesh support, and products that feature mesh networking are available on the market.
- Bluetooth mesh, officially launched in July 2017, is a highly anticipated addition to the Internet of Things (IoT) connectivity space

THREAD

- Thread was designed with the Internet's proven, open standards to create an Internet Protocol version 6 (IPv6) based mesh network, with 6LoWPAN as its foundation.
  - https://www.threadgroup.org/

OPENTHREAD

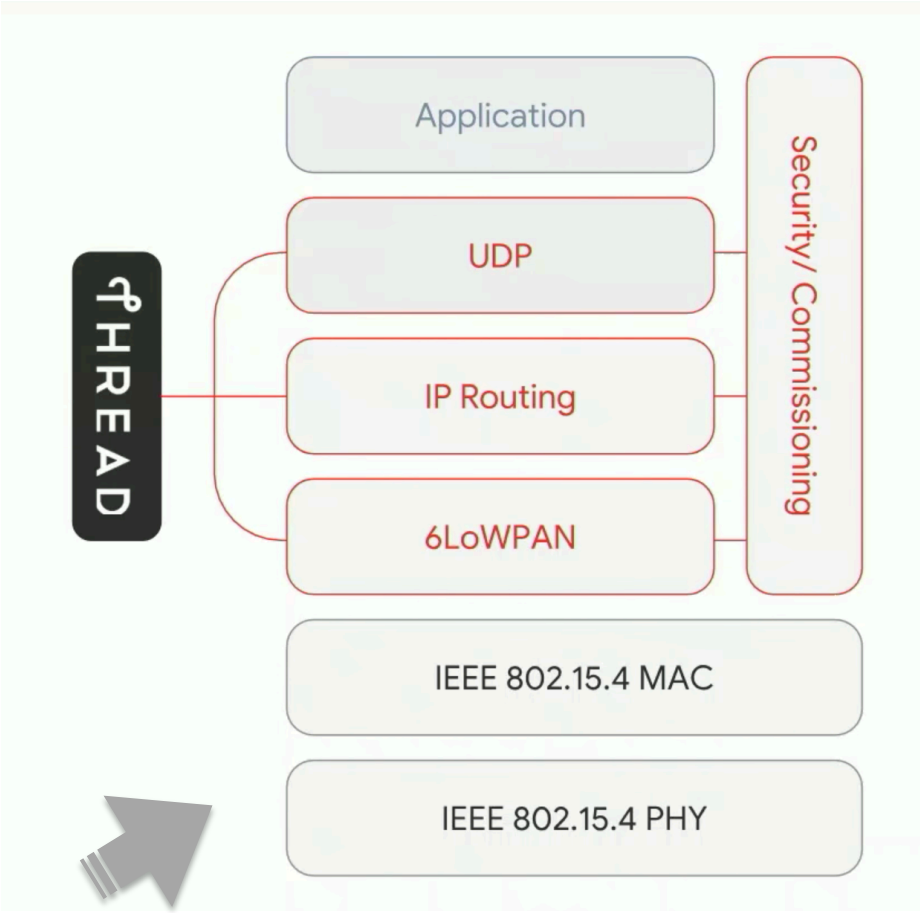
released by Google

https://openthread.io/guides

Thread

- Thread has been described as a "low-latency low-power mesh network protocol". Its specification defines a communication protocol for home applications.

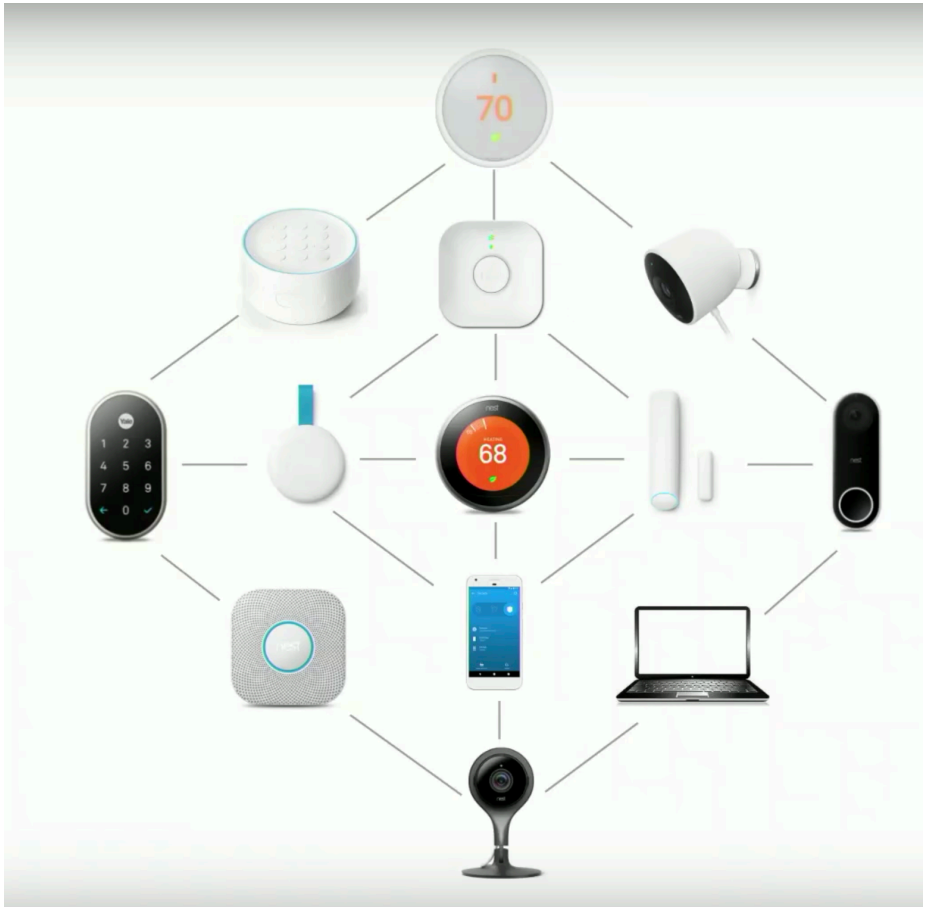
- Thread's primary features include:
  - **Simplicity** — Simple installation, start up, and operation
  - **Security** — All devices in a Thread network are authenticated and all communications are encrypted
  - **Reliability** — Self-healing mesh networking, with no single point of failure, and spread-spectrum techniques to provide immunity to interference
  - **Efficiency** — Low-power Thread devices can sleep and operate on battery power for years
  - **Scalability** — Thread networks can scale up to hundreds of devices



OpenThread

- OpenThread released by Google is an open-source implementation of the Thread networking protocol.
  - https://github.com/openthread

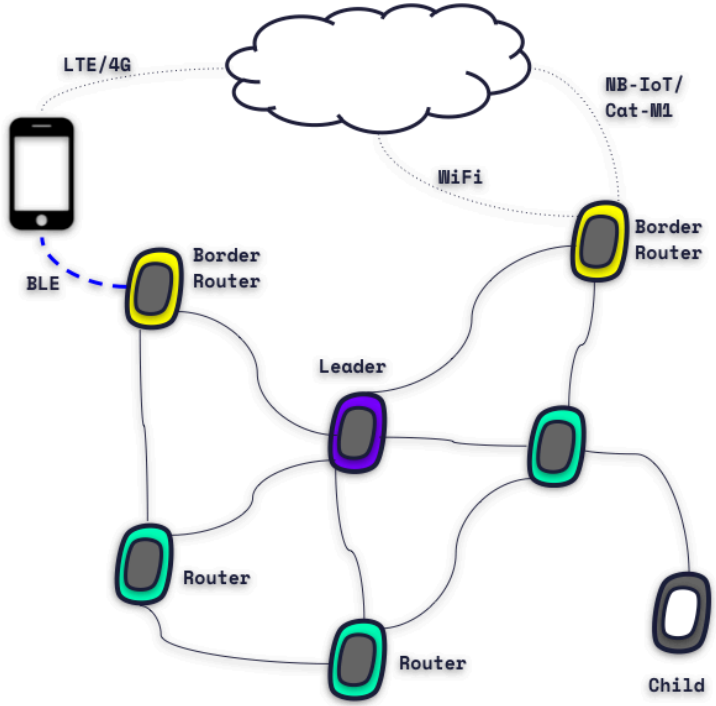
- Google Nest has released OpenThread to make the technology used in https://nest.com/ broadly available to developers to accelerate the development of products for the connected home.



Widely available

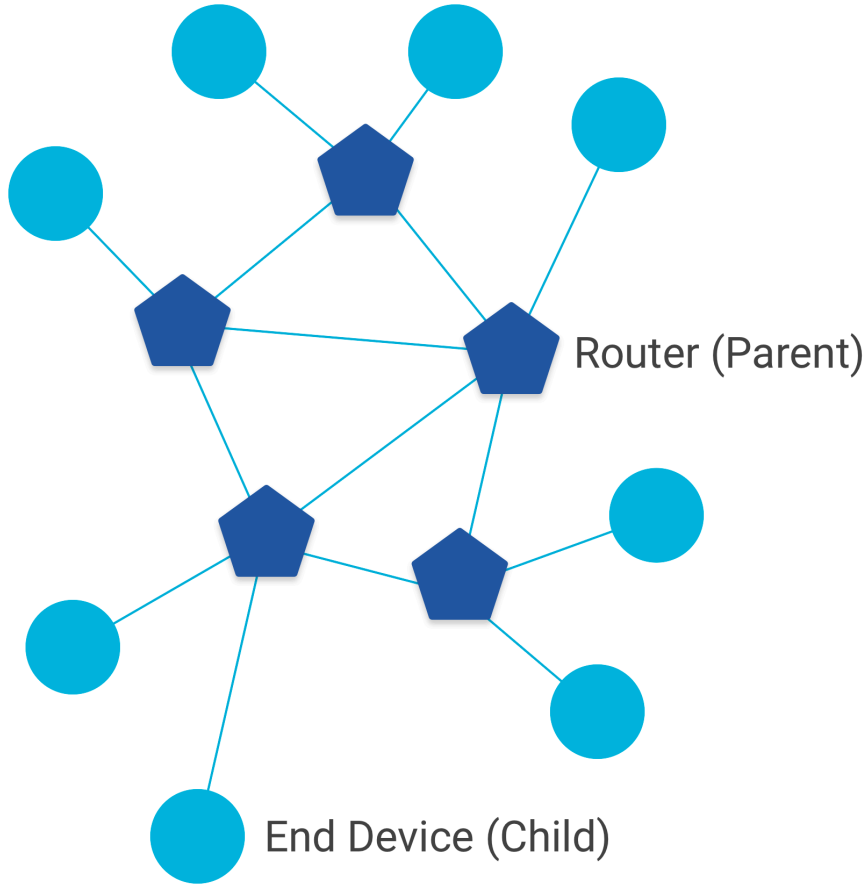
The Pymesh LoRa Mesh is implemented using OpenThread.

- https://docs.pycom.io/pymesh/lora-mesh/



Node Roles and Types

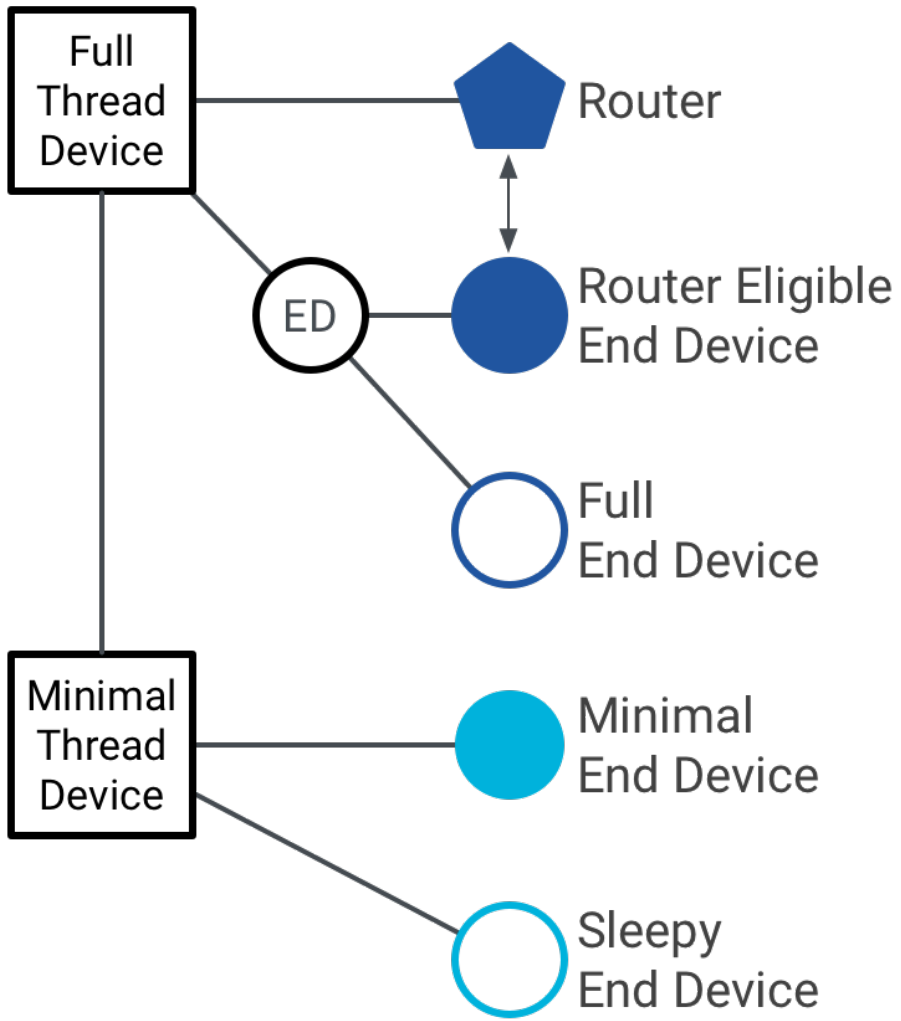
- In a Thread network, nodes are split into two forwarding roles:
  - Router: a node that:
    - forwards packets for network devices
    - provides secure "commissioning" services for devices trying to join the network
    - *always keeps its transceiver enabled*
  - End Device (ED): a node that:
    - communicates primarily with a single Router
    - does not forward packets for other network devices
    - **can disable its transceiver to reduce power**



Commissioning => "messa in servizio"

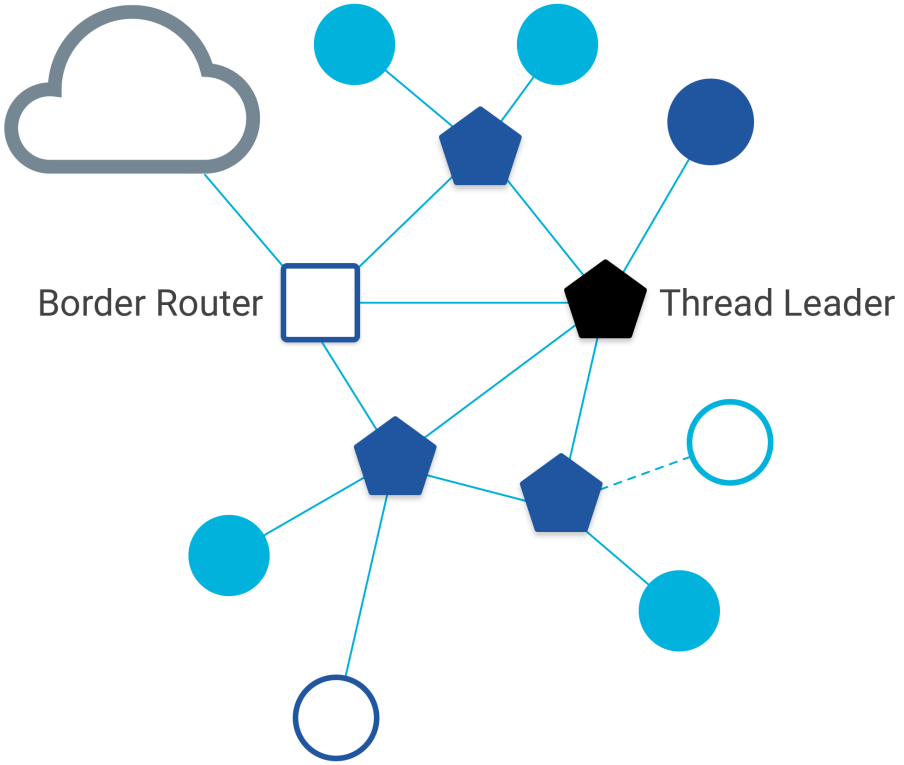
Device types

- A Full Thread Device (FTD) always has its radio on, subscribes to the all-routers multicast address, and maintains IPv6 address mappings.
  - There are three types of FTDs
- A Minimal Thread Device (MTD) does not subscribe to multicast traffic and forwards all messages to its Parent.
  - An MTD can only operate as an End Device (Child).
  - There are two types of MTDs



Other roles and types

- The **Thread Leader** is a Router that is responsible for managing the set of Routers in a Thread network.
  - It is dynamically self-elected for fault tolerance, and aggregates and distributes network-wide configuration information.
  - **Note: There is always a single Leader in each Thread network partition.**
- A **Border Router** is a device that can forward information between a Thread network and a non-Thread network (for example, Wi-Fi). It also configures a Thread network for external connectivity.
  - **Note: There can be multiple Border Routers in a Thread network.**



Device limits

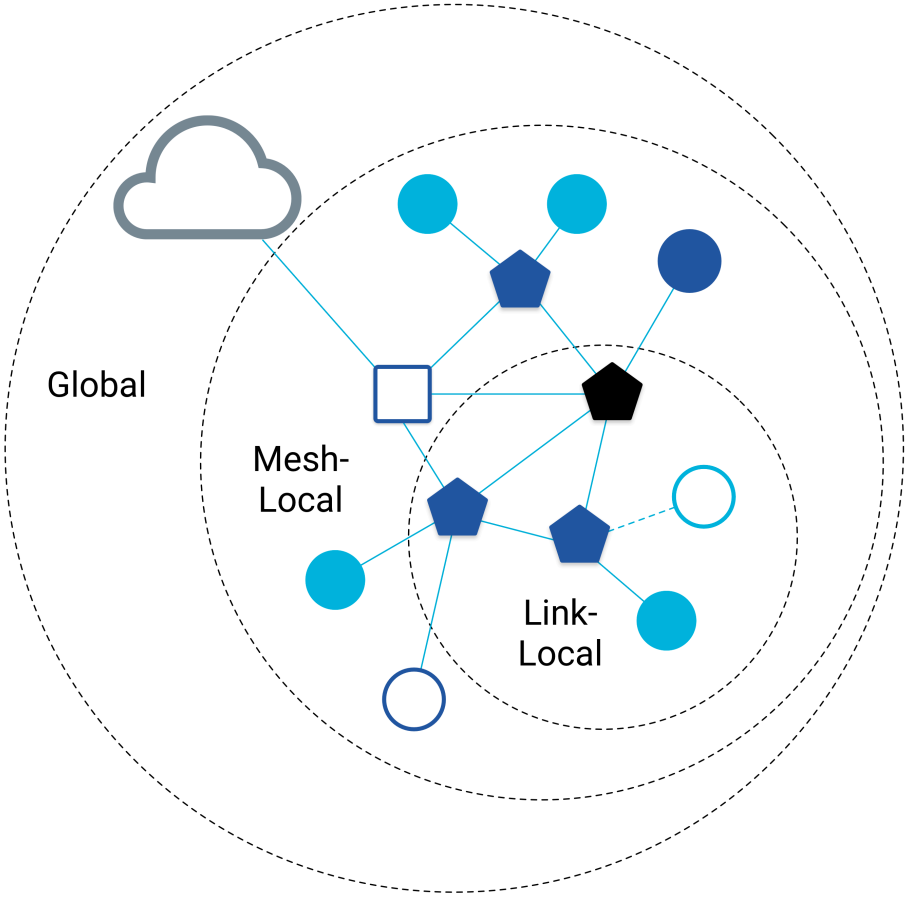
- There are limits to the number of device types a single Thread network supports.

Role	Limit
Leader	1
Router	32
End Device	511 per Router

- Thread tries to keep the number of Routers between 16 and 23.
  - For example, if a REED (*Router Eligible End Device*) attaches as an End Device and the number of Routers in the network is below 16, it automatically promotes itself to a Router.

IPv6 Addressing

- There are three scopes in a Thread network for unicast addressing:
  - **Link-Local** — all interfaces reachable by a single radio transmission
  - **Mesh-Local** — all interfaces reachable within the same Thread network
  - **Global** — all interfaces reachable from outside a Thread network



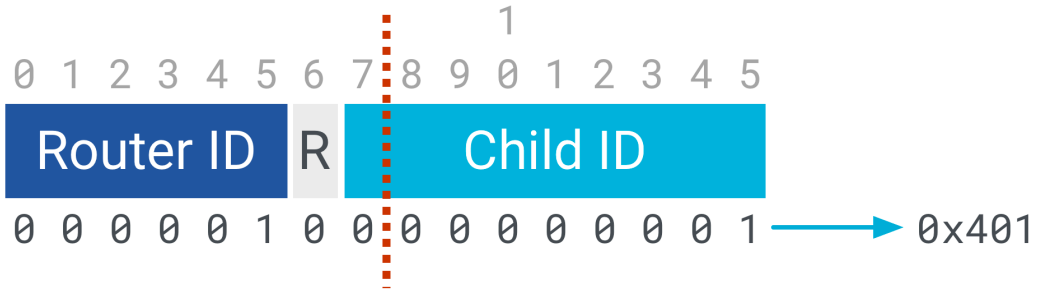
The Routing Locator (RLOC)

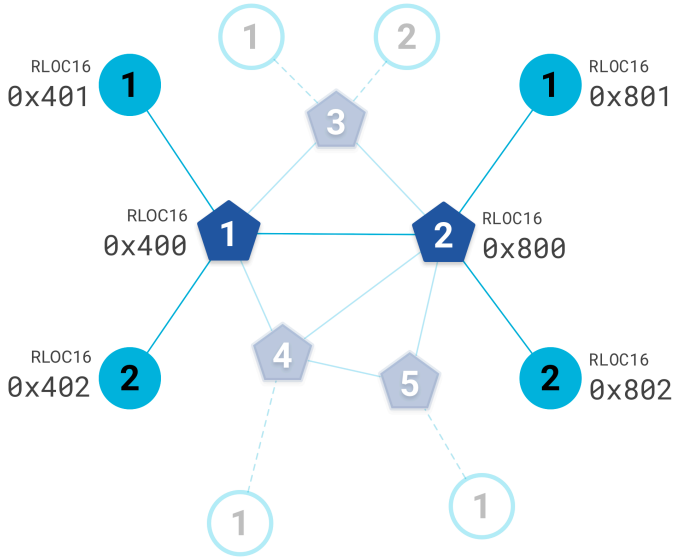
- There are multiple IPv6 unicast addresses that identify a single Thread device. Each has a different function based on the scope and use case.
- There is a common one, called the Routing Locator (RLOC).
  - The RLOC identifies a Thread interface, based on its location in the network topology.

Routing Locator (RLOC)	
Identifies a Thread interface, based on its location in the network topology.	
Example	`fde5:8dba:82e1:1::ff:fe00:1001`
IID	`0000:00ff:fe00:`*`RLOC16`*
Scope	Mesh-Local
Details	• Generated once a device attaches to a network • For delivering IPv6 datagrams within a Thread network • Changes as the topology changes • Generally not used by applications

The Routing Locator (RLOC)

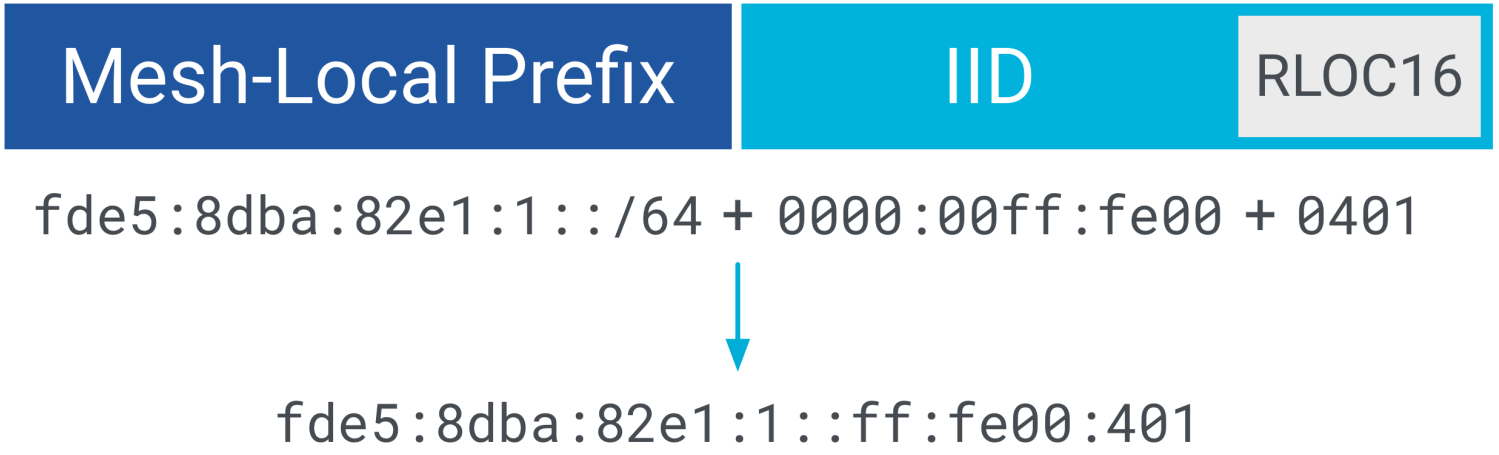
- All devices are assigned a Router ID and a Child ID. Each Router maintains a table of all their Children, the combination of which uniquely identifies a device within the topology.
- For example, consider the highlighted nodes in the topology below, where the number on a Router (pentagon) is the Router ID, and the number on an End Device (circle) is the Child ID:
  - Each Child's Router ID corresponds to their Parent (Router). Because a Router is not a Child, the Child ID for a Router is always **0**. Together, these values are unique for each device in the Thread network, and are used to create the **RLOC16**, which represents the last 16 bits of the RLOC.
- For example, here's how the RLOC16 is calculated for the upper-left node (Router ID = 1 and Child ID = 1):





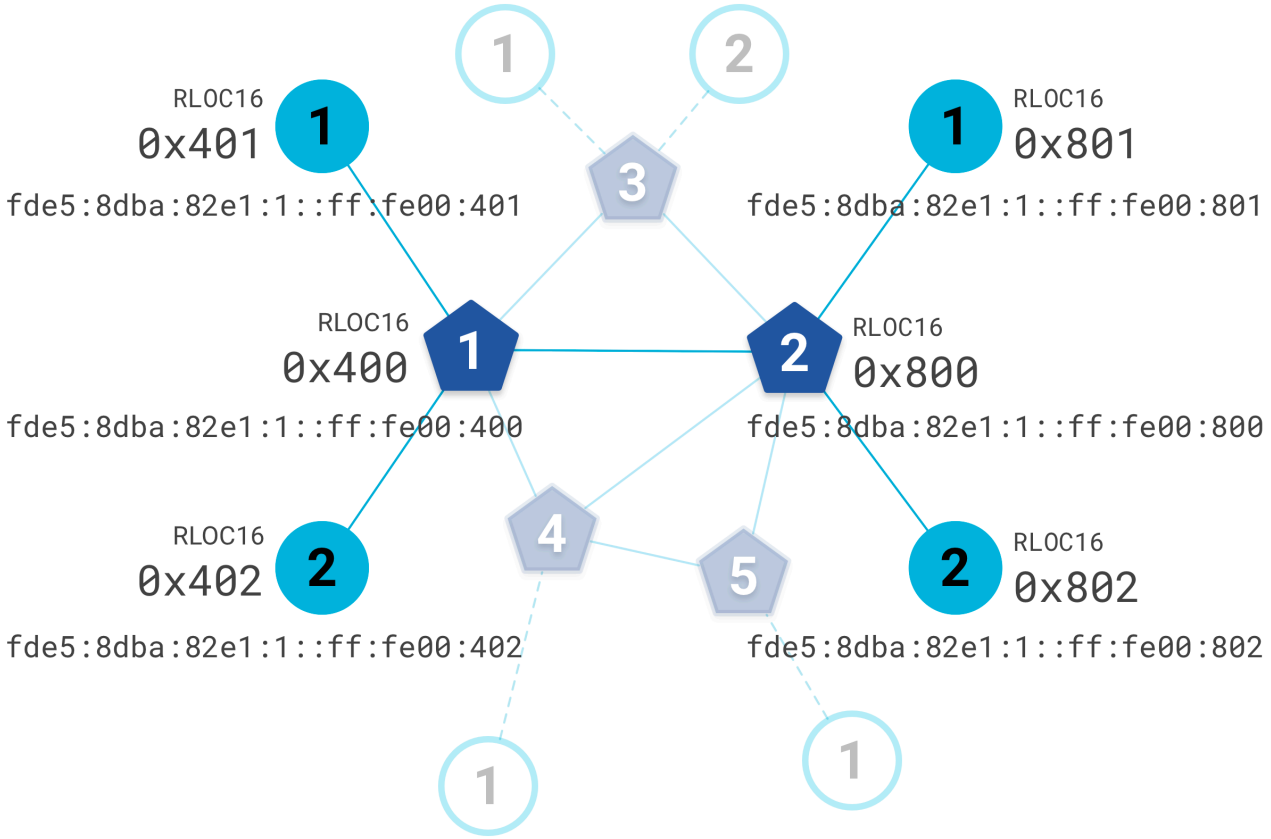
The Interface Identifier (IID)

- The RLOC16 is part of the **Interface Identifier (IID)**, which corresponds to the last 64 bits of the IPv6 address. Some IIDs can be used to identify some types of Thread interfaces. For example, the IID for RLOCs is always of the form **`0000:00ff:fe00:RLOC16`**.
- **The IID, combined with a Mesh-Local Prefix**, results in the RLOC. For example, using a Mesh-Local Prefix of fde5:8dba:82e1:1::/64, the RLOC for a node where RLOC16 = 0x401 is:



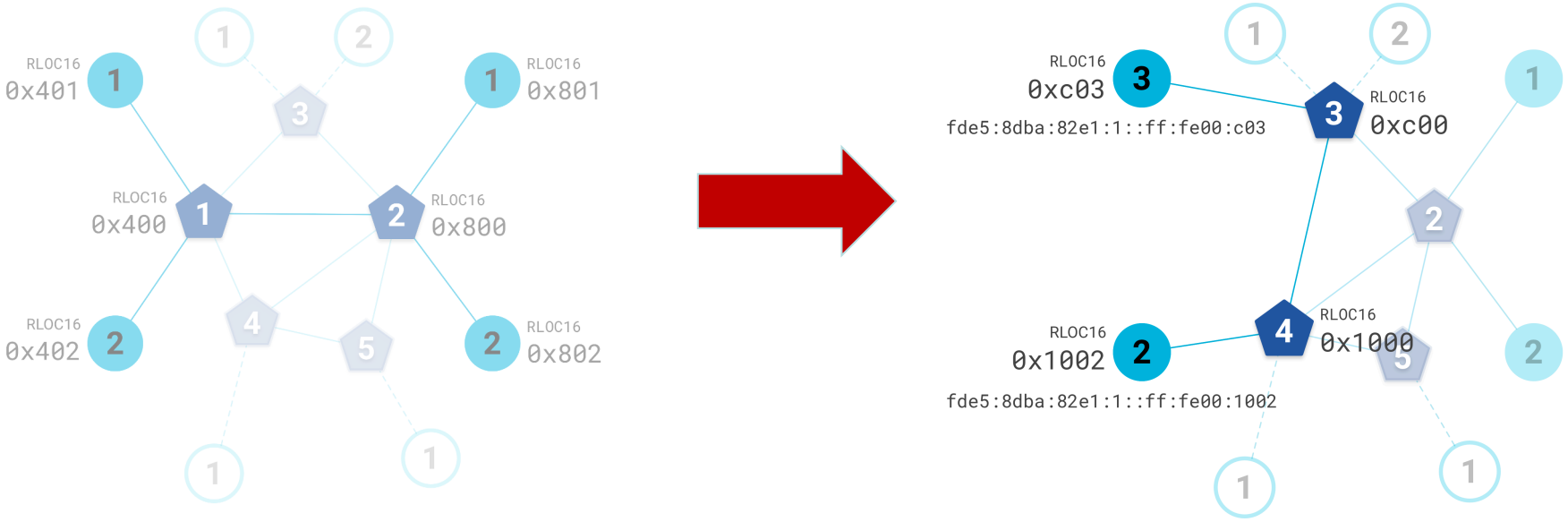
Overall example

- This same logic can be used to determine the RLOC for all highlighted nodes in the sample topology above:



RLOCs and mobility

- However, because the RLOC is based on the location of the node in the topology, the RLOC of a node can change as the topology changes.
- For example, if node **0x400** is removed from the Thread network. Nodes **0x401** and **0x402** establish new links to different Routers, and as a result they are each assigned a new RLOC16 and RLOC:



Multicast

- Multicast is used to communicate information to multiple devices at once. In a Thread network, specific addresses are reserved for multicast use with different groups of devices, depending on the scope.

IPv6 Address	Scope	Delivered to
`ff02::1`	Link-Local	All FTDs and MEDs
`ff02::2`	Link-Local	All FTDs
`ff03::1`	Mesh-Local	All FTDs and MEDs
`ff03::2`	Mesh-Local	All FTDs

Anycast

- Anycast is used to route traffic to a Thread interface when the RLOC of a destination is not known. An Anycast Locator (ALOC) identifies the location of multiple interfaces within a Thread partition. The last 16 bits of an ALOC, called the ALOC16, is in the format of **`0xfcXX`**, which represents the type of ALOC.
  - For example, an ALOC16 between **`0xfc01`** and **`0xfc0f`** is reserved for DHCPv6 Agents. If the specific DHCPv6 Agent RLOC is unknown (perhaps because the network topology has changed), a message can be sent to a DHCPv6 Agent ALOC to obtain the RLOC.

ALOC16	Type
`0xfc00`	Leader
`0xfc01` – `0xfc0f`	DHCPv6 Agent
`0xfc10` – `0xfc2f`	Service
`0xfc30` – `0xfc37`	Commissioner
`0xfc40` – `0xfc4e`	Neighbor Discovery Agent
`0xfc38` – `0xfc3f` `0xfc4f` – `0xfcff`	Reserved

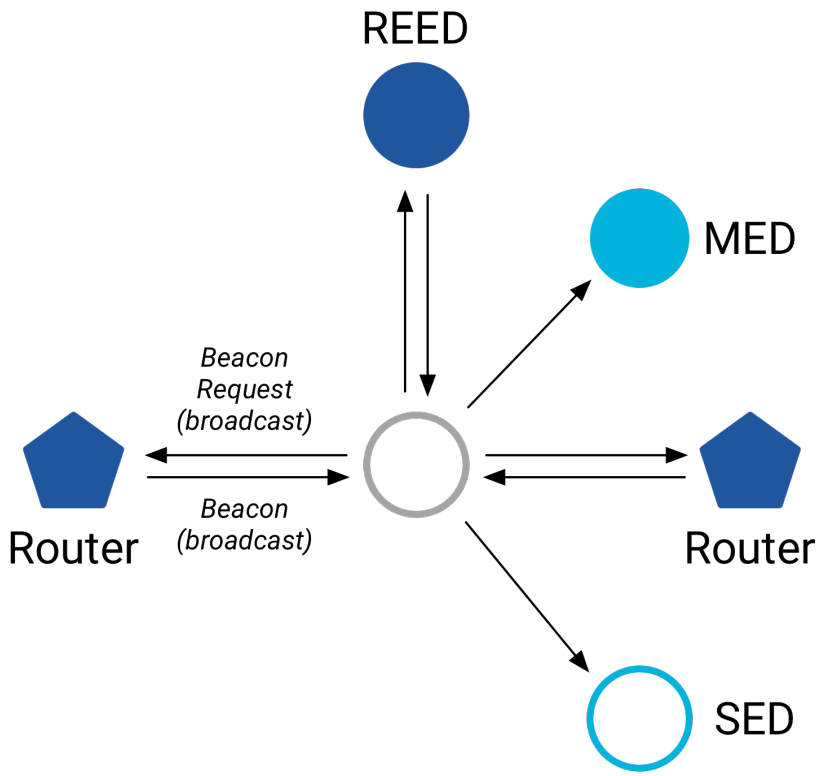
Network Discovery and Formation

- Thread networks are identified by three unique identifiers:
  - 2-byte Personal Area Network ID (PAN ID)
  - 8-byte Extended Personal Area Network ID (XPAN ID)
  - A human-readable Network Name
- For example, a Thread network may have the following identifiers:

Identifier	Value
PAN ID	**0xBEEF**
XPAN ID	**0xBEEF1111CAFE2222**
Network Name	yourThreadCafe

A Thread network

- When creating a new Thread network, or searching for an existing one to join, a Thread device performs an active scan for networks within radio range:
  1. The device broadcasts an 802.15.4 **Beacon Request** on a specific Channel.
  2. In return, any Routers or Router Eligible End Devices (REEDs) in range broadcast a Beacon that contains their Thread network PAN ID, XPAN ID, and Network Name.
  3. The device repeats the previous two steps for each Channel.
- **Once a Thread device has searched for all networks in range, it can either attach to an existing network, or create a new one if no networks are discovered.**



Mesh Link Establishment

- Thread uses the **Mesh Link Establishment (MLE)** protocol to configure links and disseminate information about the network to Thread devices.
- In link configuration, MLE is used to:
  - Discover links to neighboring devices
  - Determine the quality of links to neighboring devices
  - Establish links to neighboring devices
  - Negotiate link parameters (device type, frame counters, timeout) with peers
- MLE disseminates the following types of information to devices wishing to establish links:
  - Leader data (Leader RLOC, Partition ID, Partition weight)
  - Network data (on-mesh prefixes, address autoconfiguration, more-specific routes)
  - Route propagation: in Thread works like the Routing Information Protocol (RIP), a distance-vector routing protocol.

Create a new network

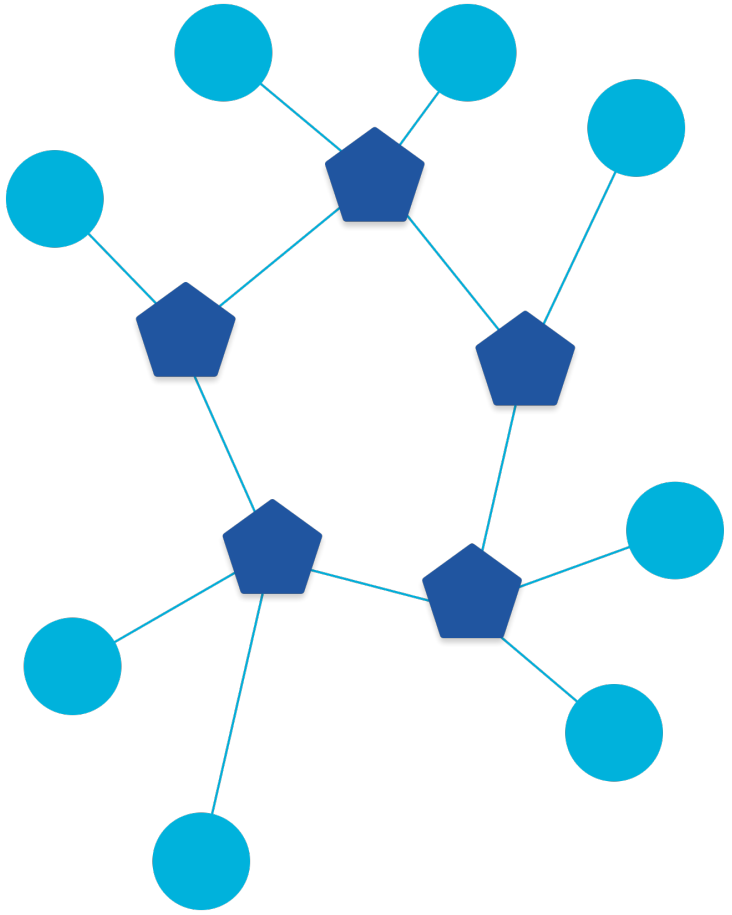
- If the device elects to create a new network, it selects the least busy Channel and a PAN ID not in use by other networks, then becomes a Router and elects itself the Leader.
  - This device sends MLE Advertisement messages to other devices to inform them of its link state and responds to Beacon Requests by other Thread devices performing an active scan.

- If the device elects to join an existing network, it configures its Channel, PAN ID, XPAN ID, and Network Name to match that of the target network via Thread Commissioning, then goes through the MLE Attach process to attach as a Child (End Device).
  - The Child sends a multicast Parent Request to all neighboring Routers and REEDs in the target network.
  - All neighboring Routers and REEDs (if the Parent Request Scan Mask includes REEDs) send Parent Responses with information about themselves.
  - The Child chooses a Parent device and sends a Child ID Request to it.
  - The Parent sends a Child ID Response to confirm link establishment.

REED (Router Eligible End Device)

Router Selection

- Routers must form a **Connected Dominating Set (CDS)**, which means:
  1. There is a Router-only path between any two Routers.
  2. Any one Router in a Thread network can reach any other Router by staying entirely within the set of Routers.
  3. Every End Device in a Thread network is directly connected to a Router.
- A distributed algorithm maintains the CDS, which ensures a minimum level of redundancy. Every device initially attaches to the network as an End Device (Child). As the state of the Thread network changes, the algorithm adds or removes Routers to maintain the CDS.



Router Selection

- **Thread adds Routers** to:
  1. Increase coverage if the network is below the Router threshold of 16
  2. Increase path diversity
  3. Maintain a minimum level of redundancy
  4. Extend connectivity and support more Children

- **Thread removes Routers** to:
  1. Reduce the Routing state below the maximum of 32 Routers
  2. Allow new Routers in other parts of the network when needed