OpenWeave

(With Demonstration using Happy Network Simulation Tool)

CS415 Wireless Networks Midsem Seminar

Topic No: - 89

Awanit Ranjan

Roll - 181CO161

National Institute of Technology Karnataka, Surathkal, India

awanitranjan.181me214@nitk.edu.in

Overview

- Thread Protocol (prerequisite required for demonstration)
- OpenWeave
- Happy Network Simulation Tool
- Experiment Demonstration with Happy
- Adding Weave to Thread Network
- Topology maintenance
- References

Thread Network Protocol

Thread which is a.k.a **WPAN** (Wireless Personal Area Network) is an **IPV6 based *Mesh Networking Protocol,*** specifically designed for the **LPD** (Low powered devices) such as **IOT devices** for the **Home Automation.**

Built on:

- **6LowPAN** as an **Adaption layer** between (Network and Data Link Layer) which defines how to send IPV6 packets over 802.15.4
- **IEEE 802.15.4** MAC/PHY ideal for low powered devices.
- No Application Layer is defined.

For providing security it uses AES(Advanced Encryption Standard).

Features of Thread:

- Secure
- Reliable
- Compatible
- Power Efficient

Note: Thread Protocol is agnostic to Application Layer and does not define an Application Layer.

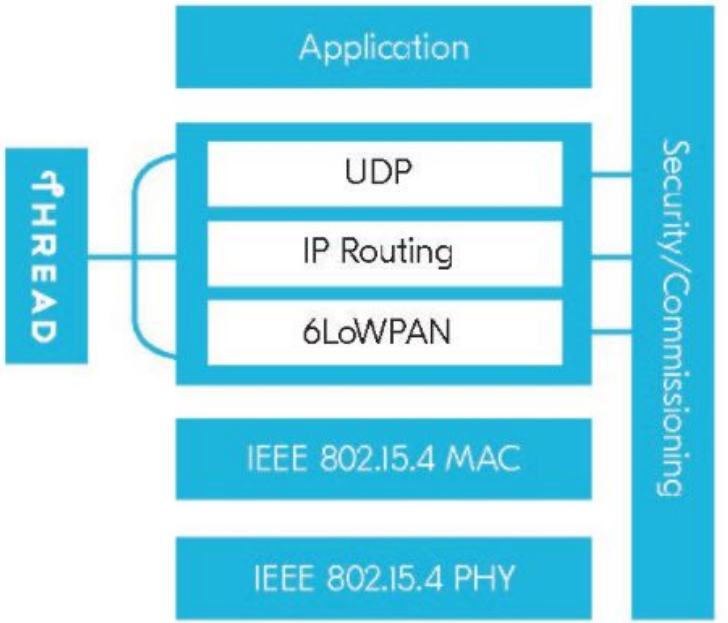


Image Credits: https://www.nordicsemi.com/products/thread

Thread Network Topology and Devices

Thread Network Topology establishes connection with other devices via Border Router. The communication within this network is based on 802.15.4 and it formed a self-healing mesh due to dynamic election of router as a Leader.

- **End Devices** : It communicates only through Thread Router i.e its parent and can't route/forward messages.
- **Thread Router :** It provides routing to network, also provides joining and security services to device trying to join the network.
- **Thread Leader :** Thread dynamically elects a leader, and this is responsible for managing a registry route or ids. If Leader fails another router assumes the roles thus there is no Single Point of Failure, providing reliability.
- **Border Router :** It provides end to end connectivity from thread network to adjacent networks. To provide resilience thread supports multiple Border Routers operating simultaneously. This provides redundant paths in and out of the network.

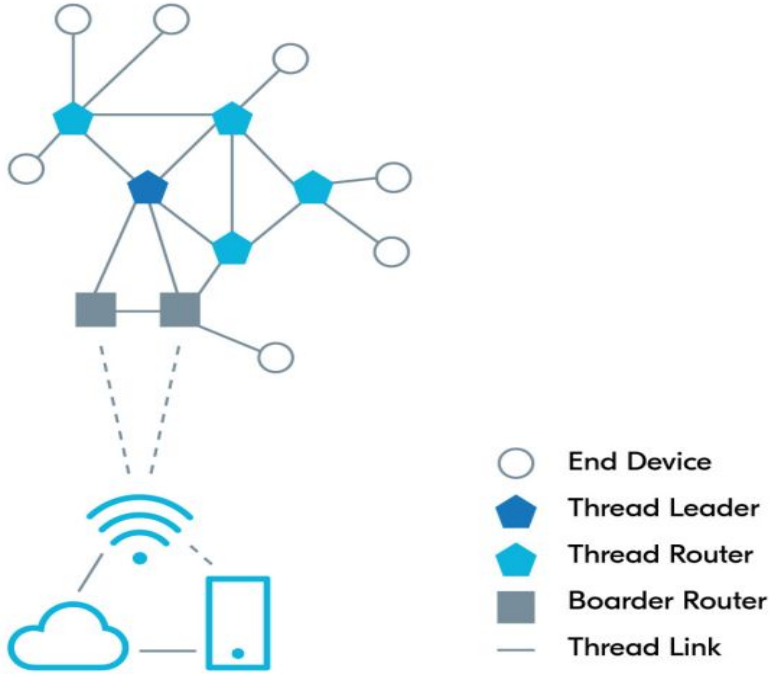


Image Credits:https://www.nordicsemi.com/Products/Thread/What-is-Thread

OpenWeave

OpenWeave is **a open-source implementation of Weave network an Application protocol stack designed for IOT devices to enable device-to-device, device-to-mobile, device-to-cloud communications.**

Weave lives on each iot devices/node in buildings/home and in weave terminology this nodes are called as **Resources.**

Weave Fabric : A collection of weave enabled devices(resources), providing a convenient way for nodes to communicate/exchange data in a secure way.

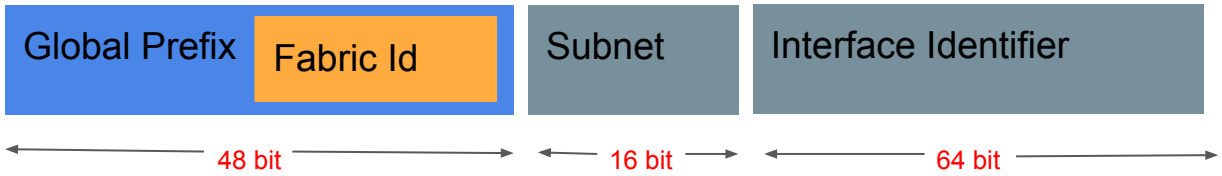
Weave is architected around IPv6 (IPv4 won't work) and it can run over any underlying technology such as Ethernet, WiFi, Thread Network..

Github Link for Openweave: https://github.com/openweave/openweave-core

More about Weave Continued...

Weave assign each resource in a weave-fabric with a IPv6 address.

This IPv6 address consist of following:



Global Prefix : All weave nodes uses IPv6 global prefix of fd00::/48.

Fabric ID : A randomly generated ID become the part of IPv6 global prefix.

Subnet : A 16 bit value determined by underlying networks.

Interface Identifier : A 64-bit value derived from node's Weave Node Id / MAC address by flipping the 7th bit of it. If MAC is 48 bit then 16-bit value **FF:FE** is inserted in between MAC to make it 48+16=64 bit and finally 7th bit is flipped to transform to Interface Identifier.

Happy Network Simulation Tool

It is a lightweight simulation tool for simulating complex network topology on Single Linux Machine without using iot devices hardware.

Note : Happy is built on Linux network namespaces, thus a Linux machine is required to use Happy.

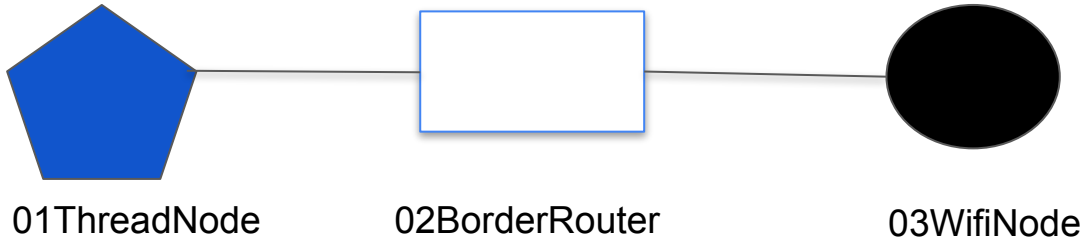
Github Link for Happy Tool: https://github.com/openweave/happy

Link for Installing Happy and configuring Happy with Weave :

https://openweave.io/codelabs/happy-weave-getting-started#1

Experiment : Demonstration of Creating a two nodes connected via Border Router.

Three Node Topology with Happy



- First, creating the three nodes:

```
$ happy-node-add 01ThreadNode

$ happy-node-add 02BorderRouter

$ happy-node-add 03WiFiNode
```

- Now let's create some networks:

```
$ happy-network-add ThreadNetwork thread
$ happy-network-add WiFiNetwork wifi
```

- For Checking the Happy state:

```
$ happy-state
```

- Adding each node to the appropriate network(s): Note : as Border Router connects two different network so it must be in both.

```
$ happy-node-join 01ThreadNode ThreadNetwork
$ happy-node-join 02BorderRouter ThreadNetwork
$ happy-node-join 02BorderRouter WiFiNetwork
$ happy-node-join 03WiFiNode WiFiNetwork
```

- Checking the happy state:

```
$ happy-state
```

Now our topology would be like below:

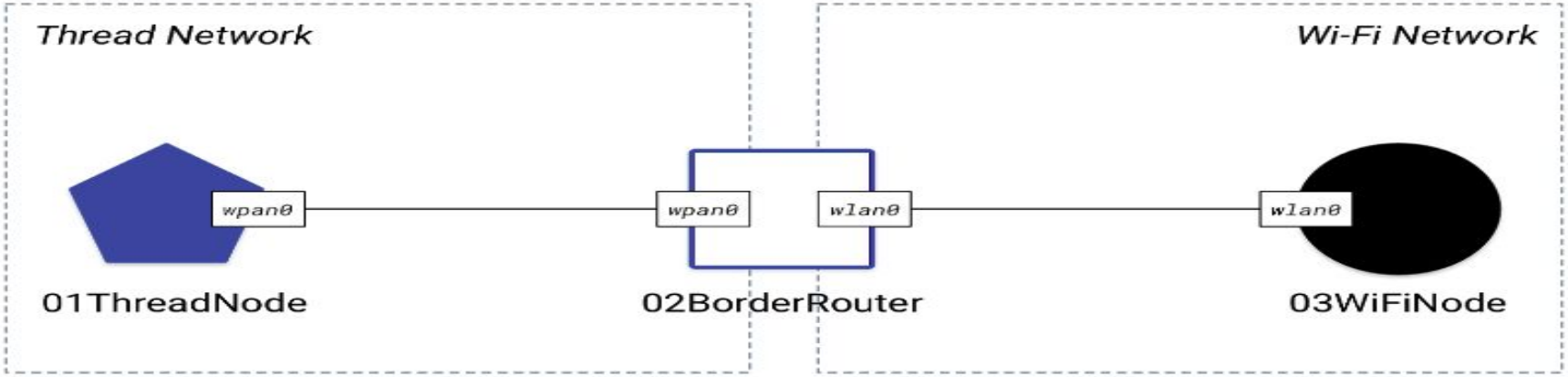


Image Credits: https://openweave.io/codelabs/happy-weave-getting-started#2

Now we have to assign IP addresses to each interfaces on each node. The best thing is that we just need to assign the IP prefix for the network and Happy automatically assigns IP address to the node's interface in that network.

.

- Assigning IP prefix for Thread Network (As the Thread protocol uses or is based on IPv6 implementation, so we add an IPv6 prefix to the Thread network)

```
$ happy-network-address ThreadNetwork 2001:db8:1:2::
```

- Similarly Assigning IP prefix for Wifi - Network, in wifi we can use both IPv4 and IPv6 so lets assign both:

```
$ happy-network-address WiFiNetwork 2001:db8:a:b::
$ happy-network-address WiFiNetwork 10.0.1.0
```

- Checking Happy state

```
$ happy-state
```

Adding Route

For each node, we have to configure:

- the nearest network gateway—in this case, **02BorderRouter** for both
- the target network—where to go after the gateway.

Thus we have :

from Source Network	to Target Network	via Gateway
ThreadNetwork	WiFiNetwork	02BorderRouter wlan0 2001:db8:1:2::/64 prefix
WiFiNetwork	ThreadNetwork	02BorderRouter wpan0 2001:db8:a:b::/64 prefix

- In Happy instead of configuring routing for each gateway we can do straightforward by network:

```
$ happy-network-route -a -i ThreadNetwork -t default -v 02BorderRouter -p 2001:db8:1:2::/64
$ happy-network-route -a -i WiFiNetwork -t default -v 02BorderRouter -p 2001:db8:a:b::/64
```

- Retrying the Ping

```
$ happy-ping 01ThreadNode 02BorderRouter
$ happy-ping 01ThreadNode 03WiFiNode
```

The both ping will work now also the IPv6 pings work! As, With forwarding on, it knows how to reach the **wlan0 or wpan0** interface. The IPv4 ping still fails, because Thread doesn't run over IPv4.

Adding Weave

Weave sits on top of the home area network and enables easier routing across the different underlying network link technologies (for example, Thread or Wi-Fi).

- Creating the Weave fabric by using **fab1** as the **Fabric ID,** & then configuring all nodes for Weave:

```
$ weave-fabric-add fab1
$ weave-node-configure
```

- Weave nodes on a Thread network need to know where to exit that network. A Weave network gateway—typically on a Thread Border Router—provides this functionality.(This is analogous to the **happy-network-route** , but specific to Weave fabric routes.)

```
$ weave-network-gateway ThreadNetwork 02BorderRouter
```

- Checking the Happy State :

```
$ happy-state
```

- **To get more information on the Weave fabric :**

```
$ weave-state
```

- We can also save the Happy Layout for later use (say saving file's name as ***filename***):

```
$ happy-state -s filename.json
```

- Once saved Topology can be deleted :

```
$ happy-state-delete
```

- To load the saved Topology (if Topology includes weave) :

```
$ weave-state-load filename.json
```

- To load the saved Topology (if Topology does not includes weave) :

```
$ happy-state-load filename.json
```

- Checking the happy state the topology will be loaded successfully:

```
$ happy-state
```

Topology Maintenance

References:

https://en.wikipedia.org/wiki/Thread_%28network_protocol%29

https://openweave.io/

https://openweave.io/guides/weave-primer

https://github.com/openweave/openweave-core

https://en.wikipedia.org/wiki/Weave_(protocol)

https://www.youtube.com/watch?v=cV5ktpaJv5Y

https://www.youtube.com/watch?v=-mcOee7AJPc

https://www.quora.com/What-is-6LoWPAN?share=1