Overview

Wireless devices in the network communicate using this custom RF protocol layer. It guarantees reliable transmission of variable size packets to end devices, while being optimized to allow end devices to use as little power as possible. All the while, it ensures that the network remains secure at all times.

It’s primarily designed to work on Silicon Labs chips, atop the RAIL radio abstraction layer; but it should be supported on other devices as well.

General Operation

End devices can be either regular (powered,) which can receive data at any time; and sleepy devices, which turn off their radio receivers to save power. Sleepy devices work like regular device nodes, but may perform more aggressive power saving. This means the coordinator is responsible for buffering packets destined for the device until its next check-in period.

All packets sent over the air are encrypted using an unique session key, which is negotiated when a device associates to the network. Additionally, multicast and broadcast packets are encrypted as well with a shared network key, which is automatically re-generated during the lifetime of the network.

Topology

The network operates as a standard star topology; that is, all devices communicate directly with the coordinator, which sits at the center of the network. Nodes may directly communicate with one another, if they are in range of each other. At this time, relay mode through the AP is not supported.

In the future, the network may support a tree-like structure, with powered repeater nodes that forward packets to the central network coordinator.

There is no fixed limit on the maximum number of nodes, other than that the coordinator will need to keep track of state information for each node; and that operations such as group key ratcheting require communication with every single node to distribute new key material.

Node Addresses

Each device has an unique EUI-64 address it uses when communicating over the air. During association, the device receives a “short” 16-bit identifier instead, which is used as a shorthand in the over the air protocol. These 16-bit addresses are the link layer addresses, which may change

for the same device between associations; but they are guaranteed to remain constant for a given device for as long as it's associated with the network.

Addresses in the range of `0xFE00` - `0xFFFF` are reserved for use as multicast/broadcast addresses, and for internal network use. This still leaves roughly ~65K unique node addresses available for use.

Beacons

Coordinators regularly broadcast beacons, which indicate the availability of the network. Networks are identified by a unique 16 byte quantity (such as a UUID) which is programmed into nodes. The beacon in turn contains the short address of the coordinator, and information about supported features in the network.

Additionally, beacons contain notification flags (similar to 802.11 DTIM) for buffered packets for devices in deep sleep mode.

By default, the coordinator emits a beacon every 2.5 seconds.

Security

All messages passed over the network, with the exception of beacon frames, are encrypted. Unicast messages use per node keys, whereas multicast and broadcast keys use per network keys.

All coordinators are provisioned with a public/private keypair. When a device is provisioned onto the network, it receives hashes of all public keys in use by coordinators on the network, which is used so that the device may authenticate the coordinator (which provides its public key) before commencing with the key exchange.

When a node associates to a network, the coordinator and node perform key exchange to derive a unique session key for the node/coordinator. From this session key, separate keys are derived for packets sent from the coordinator to the node, and from the node to the coordinator. Additionally, random data is exchanged for use as an initialization vector for subsequently encrypted packets. Once these keys are collected, secure communication can take place over the network.

In addition to the unicast message keys, the coordinator will share the key used to encrypt multicast/broadcast packets. The coordinator can re-issue this key periodically (called the rekey interval) to improve the network's security.

Likewise, the node or coordinator can perform a new key exchange at any point during the association.

TODO: Investigate https://github.com/jedisct1/libhydrogen performance

Association

Nodes must associate with (join) a network before they're able to pass traffic. This is done in several stages:

1. Locate best coordinator

The node will scan all supported channels for beacons from coordinators. Beacons from all coordinators with matching network IDs will be stored, and sorted according to their received signal strength.

1. Connect to coordinator

Tune to the channel of the coordinator node that was declared "best" earlier, and wait to receive another beacon frame. If this times out, return to step 1.

Otherwise, send an association request (containing our full EUI-64 node address and supported protocol version and security modes) to the coordinator, and wait for a response. This response will indicate a temporary short node address to use for MAC frames during the association process. (A separate, smaller namespace for short node IDs during association helps guard against denial-of-service attacks against a single coordinator from exhausting node IDs shared across a larger network.)

1. Begin Key Exchange

Coordinator provides its public key, random data, and more network information. The public key is hashed and compared against the allowed list, and if it is allowed, the association continues. Otherwise, it's aborted, an error is logged, and another coordinator is attempted.

This message also contains information about the network's authentication mode. If no authentication is implemented, skip to (5).

1. Authenticate to network

If the network has security (as indicated by the coordinator's info message) the appropriate authentication is performed. This will consist of one or more round trips to the coordinator to complete some sort of challenge/response protocol. Devices will normally authenticate using their public/private keypair (which was previously provisioned onto the network) in a challenge/response protocol, possibly also including the node's EUI; however, for in-band pairing, an authentication mechanism using a pre-shared key (pairing code) based around ECJ-PAKE is available as well.

Once the authentication process is complete, the coordinator will send to the node either an "auth success" or "auth failure" message. On success, go to (5) to continue association; otherwise restart at (1).

1. Complete Association

When both sides have exchanged session encryption keys, secure communication can commence. The coordinator will respond to the client's key exchange message with an encrypted acceptance message, which provides the client information such as its final short node address (for use in MAC header,) supported power saving modes, and so forth.

The client then responds with an acknowledgement, using its new address, and makes requests for any desired power saving options such as request buffering.

At this stage, the node is considered associated to the network, even if any additional requests from the node are still outstanding (or fail down the line.) Devices remain associated for some time period N after the last successful packet exchange between the coordinator and node. If no packet is received from the node after a period of N, the coordinator will consider the node dead and forcibly disassociate it.

The interval N is configurable, and can vary per device, based on their desired power use.

Once associated, either the coordinator or end node may send a disassociation packet to terminate the session cleanly. Immediately after the disassociation is acknowledged by the coordinator (or device, for a coordinator-initiated disassociation) the disassociation is processed.

When a disassociation takes place, any encryption keys are zeroed and discarded, as well as any buffers and other information is also flushed. This means that packets destined for a sleeping node that missed its check-in window will be lost.

Device Types

Two device types exist: always on, and periodic devices. This difference has to do with the power saving abilities of the device: an always on device must *always* be listening, i.e. always able to accept and receive packets destined for it. By default, a device is assumed to be always on; if it wishes to operate in periodic mode, it must negotiate this once it's associated to the network.

This means that an always on device cannot make use of lower power states, where the radio is fully powered off for a period of time; therefore, this is more suited for permanently powered devices.

On the other hand are periodic devices. These are only able to accept direct messages during certain time windows. At all other times, any packets meant for the device are buffered by the coordinator, and will be retrieved by the device at a later time. The device will wake up periodically (at a multiple of the beacon interval, negotiated during association) to receive the coordinator beacon frames. These frames indicate which periodic devices have pending traffic: devices may then wake up, download buffered frames, and process them as appropriate.

Periodic devices may also wake up at any other time, and explicitly poll the coordinator for pending traffic. This gives the device explicit control over how it implements power saving modes, with the only constraint being a maximum sleep interval, after which the network coordinator assumes the device has gone away.

Revision #9
Created 14 August 2022 17:26:46 by Tristan
Updated 25 October 2022 00:03:57 by Tristan