Tiny Home Area Network (TinyHAN) Protocol Specification

(C) 2013-2014 Mike Stirling

http://www.tinyhan.co.uk/

V0.04

2014-11-17

Table of Contents

License

The TinyHAN software stack is licensed under the Apache 2.0 License. You can obtain a copy of the license form the following URI:

http://www.apache.org/licenses/LICENSE-2.0

Revision History

2014-08-28	Initial revision
2014-08-29	Removed wallclock time from beacons in favour of separate message. Added placeholder for Si446x (RFM24W and RFM26W). Added note on MQTT-SN implementation.
2014-10-16	Added new packet types for "TinyHAN Application Protocol", plus a specific one for MQTT-SN and a reserved type for extension purposes.
2014-11-06	Reorganised packet type assignments and removed "Ping" in favour of "Poll", used only by a client wishing to poll its coordinator for data. Added concept of heartbeat interval. Clarified procedure for handling messages from unregistered addresses. Added note on sleeping node timing. Removed cross-network messaging capability.
2014-11-17	Clarified procedure for deferred transmissions.

TinyHAN Lightweight Home Area Network

This document proposes simple PHY and MAC layers for the implementation of a non-mesh Low Power Wireless Personal Area Network (LoWPAN) intended for domestic applications. Although loosely based on 802.15.4 it is designed to be implemented on low cost radio hardware such as HopeRF RFM23B/22B and RFM69W.

The protocol was originally designed to perform MAC and transport layer functions for a simple network based on MQTT-SN, but the layered approach allows for applications using other higher-layer protocols such as 6LoWPAN. A very lightweight binary application layer is also proposed and may be preferable for applications where node resources are particularly constrained.

Topology

Devices may be built with either node-only or full coordinator capabilities, at least one of the latter being required within a network. If multiple coordinators are operating within a single network, for example for range-extension, then they must use a shared address allocation table - the mechanism for this is beyond the scope of this specification.

The MAC can support peer-to-peer operation but does not provide neighbour discovery services, which would need to be handled by the upper layers. In a typical deployment the topology would be client-server with the hub acting as both network coordinator and gateway.

All devices have a 64-bit Unique Unit Identifier (UUID) which is conveyed to the coordinator during registration.

Security

Consideration has been given to the addition of authentication and encryption extensions, but these are not specified in this version of the protocol. A whitelist may be used on the coordinator to reject registration requests from unknown devices.

Physical (PHY) Layer

The PHY is based on the mandatory FSK profile from 802.15.4g and was developed using HopeRF RFM23B radios (SiLabs Si4431). Other radios are likely to be interoperable. Radio characteristics are as follows:

- Operating frequency range: 863 MHz to 870 Mhz
- 200 kHz channel spacing (channel 0 centre is 863.125 MHz)
- GFSK modulation (BT=0.5)
- 50 kbps
- +/- 25 kHz deviation (M=1)

The following services are provided by the physical layer for use by the MAC:

- Configuration of transmitter power
- Configuration of transmit preamble length (4 to 1000 bytes)
- Transmission of packets including calculation of CRC
- Reception of packets including validation of CRC and RSSI measurement
- Listen/Standby mode selection
- Energy detection (returns current channel RSSI)
- Clear channel assessment (RSSI below some variable threshold)

A physical layer Protocol Data Unit comprises the following fields. All bytes are transmitted MSb first and all multi-byte values are transmitted MSB first (big-endian).

	Preamble	Sync	~~PHY Flags~~	Payload length	Payload	CRC
Bytes	>=4	2	~~1~~	1	N	2
	0101.....01	0x2DD2	~~0x00~~			
CRC			~~TBC~~			

The sync word is deliberately chosen to be different to the 802.15.4g one to avoid interference with compliant 802.15.4 MACs. ~~The flag byte is reserved and should be set to 0x00.~~

The CRC here is provided by the physical layer hardware and uses polynomial 0x1021 with an initial value of 0 (RFM22B/RFM23B format). It is calculated over the fields marked above. This is subject to change (test platform is built using RFM23B only). To ensure maximum compatibility between different radios the most likely change is to omit CRC checking in the radio and to incorporate a common CRC at the MAC layer.

Media Access Control (MAC) Layer

The MAC is designed for compatibility with a range of low-cost sub-GHz radios. In particular the HopeRF RFM22B/23B and RFM69xW range are supported.

All nodes are assigned a 64-bit unique identifier.

The payload always begins with a MAC layer header of the following format:

	Flags	Network ID	Destination Address	Source Address	Sequence Number	Payload
Bytes	2	1	1	1	1	N
Optional encryption TBC						

Proposed encryption extensions would operate over the highlighted fields. The flags field would be sent in the clear to allow the receiver to determine the encryption mode.

A proposed authentication mode would place a Message Authentication Code (MAC) between the sequence number and the payload.

Network ID is assigned at random by the coordinator and is used to distinguish between multiple networks operating within radio range (and to force reorganisation if the coordinator is restarted). 0xFF is a reserved value used to indicate "all networks" for the purpose of initial registration only..

8-bit short addresses are assigned by the coordinator upon a successful registration request. The short address of the coordinator is obtained from a beacon request - there may be more than one coordinator servicing a given network ID (in this case multiple coordinators must share address assignments by some external means). 0xFF is a reserved value used to indicate a broadcast (if used as destination) or a message from a currently unregistered node (if used as source). **Packets with a source address or network ID of 0xFF must be ignored except by coordinator nodes in "Permit Attach" state.**

Sequence number increments for each packet transmitted by a node. The same sequence number is used regardless of packet type except for synchronisation beacons, which have their own counter. Acknowledgement packets use the sequence number of the packet being acknowledged.

The flags word is transmitted MSB first and encoded as follows:

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Version			Reserved			Reserved for crypto		DP	AR	Rsvd	Packet type				

All reserved fields shall be set to 0 and ignored by the receiver.

Version shall be 0b000

DP indicates further data pending after this frame. The receiver shall remain in listen mode for a (TBC) guard period after the end of this frame. If this frame requires acknowledgement then the listen period shall commence immediately after the transmission of the ACK.

AR indicates that this frame is to be acknowledged by the target MAC.

DP and AR bits shall be ignored on packets sent to the broadcast address.

Packet type shall be selected from one of the following:

0x00	Beacon (sync or coordinator advertisment)
0x01	Beacon Request
0x02	Poll
0x03	Acknowledgement
0x04	Registration Request
0x05	Deregistration Request
0x06	Registration Response
0x07	Reserved
0x08-0x0F	Reserved
0x10	TinyHAN Application Protocol
0x11	Reserved for MQTT-SN
0x12-0x1E	Reserved
0x1F	Reserved for extended packet types

All packets shall be transmitted using CSMA/CD except for synchronisation beacons (type 0x00), which are sent unconditionally at the start of the period indicated by the encoded timestamp.

0x00 Beacon

This is transmitted by a coordinator node at regular intervals defined by the interval and offset values contained in the beacon packet (for a Sync Beacon), or on demand using CSMA/CD in response to a Beacon Request command (an Advertisement Beacon). The start of a Sync Beacon transmission shall coincide with the start of the beacon slot represented by the packet's timestamp field.

Regular Sync Beacons are useful to reduce latency to sleeping nodes, but their use imposes stricter timing requirements on client devices. A network can operate without Sync Beacons by setting the interval to 15.

DP = 0, AR = 0

Network ID = As assigned

Destination address = Broadcast (0xFF)

Source address = Originating node address (coordinator only)

	64-bit UUID	Timestamp	Flags	Beacon Interval	Address list
Bytes	8	2	1	1	N

The UUID is the 64-bit unique ID of the coordinator node transmitting this beacon.

The timestamp field is a count of beacon slots (250 ms periods) since system startup.

Flags:

7	6	5	4	3	2	1	0
Reserved						Permit Attach	Sync

All reserved fields shall be set to 0 and ignored by the receiver.

Permit Attach shall be set to 1 only when the coordinator is accepting registration requests. Devices shall not attempt to register with a coordinator whose Permit Attach bit is clear, and coordinators shall ignore any attempt to do so.

Sync shall be set to 1 only when this packet was sent without CSMA/CD on a well-defined clock boundary. Receivers may use this packet to synchronise their clocks.

Beacon Interval:

7	6	5	4	3	2	1	0
Offset				Interval			

Interval defines the time between Sync Beacons. The time is equal to 2^(Interval) beacon slots,

where each beacon slot is 250 ms apart. If Interval is greater than 0 then Offset may be non-zero in order to shift the slot in which the beacon is transmitted. This is useful in networks where multiple synchronised coordinators are operating. Interval = 15 may be used in an Advertisement Beacon to indicate that the network does not support synchronous operation (no Sync Beacons will be sent).

Address list is a variable length array of 8-bit short addresses for which the coordinator has pending data. A device whose address appears in this list should remain awake at the end of the beacon and issue a "Poll" command in order to be sent the pending packets. This list shall only be included when the beacon is a Sync Beacon.

Receivers may align their clocks on receipt of a Sync Beacon and arrange to be listening to some or all depending on the required latency vs battery life trade-off. There is no requirement for devices to listen for beacon packets - a polling approach is also possible by ensuring that the device sends a data packet with AR=1 at regular intervals.

Receivers may align their clocks on receipt of an Advertisement Beacon but must assume a lower level of timing accuracy than for a Sync Beacon. The timestamp for an Advertisement Beacon is appropriate to the beacon slot in which the beacon is being sent even if the Beacon Interval for the network would not normally transmit a beacon during that slot.

0x01 Beacon Request

An unregistered device may broadcast this message in order to solicit a beacon from in-range coordinator nodes. Coordinators shall respond with a Beacon message with the Sync bit clear. The originating device may attempt to register with a coordinator that advertises itself with its "Permit Attach" flag set.

DP = 0, AR = 0

Network ID = Any network (0xFF)

Destination address = Broadcast (0xFF)

Source address = Unassociated device address (0xFF)

There is no payload.

0x02 Poll

This is a null packet with no effect other than to solicit an acknowledgement and subsequent transmission of pending data from a coordinator. It may be used by a client node to request data from a coordinator either asynchronously or in resposne to a Sync Beacon in which the client's address appears in the address list.

DP = 0, AR = 1

Network ID = As assigned

Destination address = Target node's unicast address (typically the coordinator's short address)

Source address = Originating node address

There is no payload.

0x03 Acknowledgement

Sent in reply to any packet received to the node's unicast short address and with the AR bit set. The sequence number is set to the same value as the packet being acknowledged.

DP = 1 if the originating device has pending data to pass to the destination

AR = 0

Network ID = As assigned

Destination address = Source address of packet being acknowledged

Source address = Originating node address (destination address of packet being acknowledged)

There is no payload. *Optional RSSI e.g. to enable power control?*

Acknowledgements must not be sent for packets arriving to the broadcast address.

0x04 Registration Request

Sent by an unregistered device wishing to register with a coordinator, having previously identified the coordinator via receipt of a beacon. Devices may attempt registration with any coordinator having its Permit Attach bit set.

DP = 0

AR = 0

Network ID = As in beacon

Destination address = Short address of coordinator from beacon

Source address = Unassociated device address (0xFF)

The payload is as follows:

	64-bit UUID	Flags
Bytes	8	2

64-bit UUID is that of the originating node.

Flags are mainly reserved:

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved											Sleepy	Heartbeat Interval			

If the Sleepy bit is set then this node is assumed not to be listening except (perhaps) during beacon slots. In this case the coordinator will not attempt to forward data to it immediately, but will instead buffer it for deferred delivery.

Heartbeat Interval describes the maximum length of time (2^n seconds) that the device is allowed to wait between transmissions. After registration the coordinator must receive a transmission from the device at least this frequently otherwise it is assumed to have gone away (the coordinator shall not attempt to ping the missing device)..

If a device was previously registered with a particular coordinator then it shall only attempt association with the same coordinator unless the previous pairing is removed by some user-initiated process. This is to ensure that a network can recover itself in the event of a coordinator going offline. An unregistered device with no saved pairing shall attempt to attach to the strongest available coordinator. If no coordinator is available then the device shall try again later.

0x05 Deregistration Request

Sent by a registered device that wishes to leave the network for some reason.

DP = 0

AR = 0

Network ID = As assigned

Destination address = Short address of coordinator

Source address = Originating node address

The payload is as follows:

	64-bit UUID	Reason
Bytes	8	1

Reason shall be one of the following:

0	User request
1	Power down (e.g. battery exhausted)

0x06 Registration Response

Transmitted by a coordinator in response to a Registration Request or Deregistration Request, or as required (e.g. for administrative purposes).

DP = 1 if in response to a Registration Request and the coordinator has pending data to pass to the destination

AR = 0

Network ID = As assigned

Destination address = Target short address (for deregistration or address updates/collision resolution), or broadcast address (for registration)

Source address = Originating node address (must be a coordinator)

	64-bit UUID	Allocated short address	Status/Reason Code
Bytes	8	1	1

For registration the 64-bit UUID is that of the node being addressed, otherwise it is set to 0. Only a

node having the corresponding UUID shall act upon a broadcast registration response.

Short address is allocated by the coordinator or is set to 0xFF if the node is being deregistered or a registration request was rejected.

Status code shall be one of the following:

0	Success (attach or detach requested by the remote node)
1	Access denied (UUID is blacklisted)
2	Network full (addressed coordinator has no free client slots)
3	Shutdown pending (unsolicited - addressed coordinator is flushing all connected nodes)
4	Administrative disconnect (unsolicited - coordinator is dropping a specific node for administrative reasons)
5	Short address not registered (unsolicited - e.g. packet received after heartbeat expiry)

A node receiving this message either via broadcast or unicast with the assigned short address equal to its own but with a mismatching UUID (including UUID 0) must immediately return to unregistered state and attempt to reassociate with the network. *(NOTE: Possible DoS vector?)*

In the event that a coordinator receives a packet from a node using an unregistered short address this message will be unicast to that address to effect a disconnection (status = 5). Clients disconnected in this way may immediately attempt reconnection by the usual method.

0x10-0x1F Data

This is a general purpose data packet. The payload is passed verbatim to the application layer, with the least significant nibble used for application selection (analogous to UDP port number).

DP = 1 if another packet follows, else 0

AR = optional

Network ID = As assigned

Destination address = Target node's unicast address

Source address = Originating node address

Timing

The maximum length of a packet (in bytes) is the sum of:

- 4 bytes preamble (assuming minimum length)
- 2 bytes sync word
- 1 byte length
- up to 255 bytes payload
- 2 bytes CRC

Total = 264 bytes, 2112 bits.

With the specified PHY parameters of 2FSK at 50 kbps this will take 42 ms to transmit (plus transmitter startup delay and any delays due to CSMA/CD back-off).

An acknowledgment has a payload length of 6 bytes and therefore a total length of 120 bits (2.4 ms).

The timeout for acknowledgement of a packet with the AR bit set needs to consider the above transmit times, processing overhead at the remote node, plus delays due to channel contention. 250 ms is suggested, simply because this is equal to the beacon slot period, and is reasonable given the above numbers. A more detailed analysis may allow for optimisation of this value.

Sleeping Nodes

A sleeping client is one which, for reasons of conserving power, leaves its receiver switched off for some of the time. A sleeping node can never be a coordinator. The receiver is assumed to be enabled and therefore the device contactable under one of the following conditions:

1. During the time of an expected Sync Beacon
2. Immediately after transmitting a packet or receiving a unicast packet with the Data Pending (DP) bit set.

It is not mandatory for a sleeping node to listen for Sync Beacons even on a synchronous network, nor is it mandatory for a coordinator to transmit Sync Beacons at all. If Sync Beacons are in use then a sleeping node may detect that a coordinator has a packet pending for it by looking for its short address in the Sync Beacon Address List. If its short address is present then the sleeping node shall transmit a Poll message and continue processing as in case 2.

After transmitting any packet a sleeping node shall switch to receive mode for a period of TBC ms (say 10?) before returning to sleep. If a packet is received during this time then the receiver shall remain on until reception is complete. Iff the received packet is addressed to the sleeping node and the Data Pending bit is set then the receiver shall once again remain on for TBC ms, otherwise the receiver shall be turned off immediately.

A coordinator shall attempt to send a pending packet to a sleeping node immediately in response to an incoming packet with AR=0, or immediately following the Acknowledgement to an incoming packet with AR=1. In the latter case the Acknowledgement shall be transmitted with AR=1. An outgoing (pending) packet with AR=1 MUST remain pending until an Acknowledgement has been

received from the sleeping node, or until its validity period expires.

Implementation

Power state management is implemented in the MAC to avoid duplication of code in the PHY drivers. However, accurate and fast timing of the listen phase is essential and may be supportable by hardware in some radios. PHYs must therefore provide the following power control interfaces:

- phy_listen();
- phy_standby();
- phy_delayed_standby(uint16_t us);

The first two calls cause an immediate state transition. The last call is used to return the radio to standby after the specified delay. It may be reinvoked, in which case the new timeout supersedes the previous one, or it may be overriden by a call to phy_listen or phy_standby.

The MAC calls phy_delayed_standby after transmitting a packet. Upon receipt of a packet the data pending bit is checked and, if it is set, phy_delayed_standby is called again in order to extend the listen period. If the data pending bit is not set then this marks the end of the transaction and phy_standby is called to go immediately to standby.

PHY drivers are expected to enter listen state automatically after transmitting a packet. Calls to phy_listen do not occur in the MAC except during initialisation for a non-sleepy node.

Deferred Transmissions

Transmitted packets are stored in the MAC for (possible) later transmission under the following circumstances:

- Acknowledgement is requested
- The destination node is sleepy

The MAC send function allows a callback to be specified which will be invoked either upon successful transmission of the packet, or after giving up. The various operating modes are outlined below:

Node Type	No Ack Requested	Ack Requested
Not Sleepy	Send immediately Packet not stored Callback (success) immediately	Send immediately Packet stored Retry immediately on ACK timeout (up to retry limit) Callback (success) on ack receipt Callback (fail) on retry limit exceeded
Sleepy	Deferred (packet stored) Packet sent after node calls in Callback (success) once sent Callback (fail) if validity period expires	Deferred (packet stored) Packet sent after node calls in Retry on next call-in if ACK timeout (up to retry limit) Callback (success) on ack receipt Callback (fail) on retry limit exceeded or validity period expires

The send function may return a negative error code in which case the callback will not be invoked. This occurs if a packet is already pending.

CSMA/CD

FIXME: Not yet implemented. To avoid blocking in the tx_packet function this is likely to need to be integrated with deferred transmits in some way, e.g. by having the PHY execute a callback when the channel is clear.

Appendix A - Native Packet Formats

This appendix describes the (de-)modulation and packet handling capabilities of various common radio transceivers, and their applicability to a TinyHAN implementation. By following these guidelines it is expected that devices utilising different radios would be able to interoperate.

This section to be completed.

Si4420 (RFM12B)

Has no packet engine. 16-bit sync word which must start 0x2Dxx (MSB first). Implementations must process the full packet on the host MCU.

Si4431/4432 (RFM23B/RFM22B)

Bit order: MSB first (LSB first optional)

	Preamble	Sync	TX Header	Packet Length	Data	CRC
Bytes	1-512	1-4	0-4	0-1	N	0-2
	0101.....01					
Manchester						
Whitening						
CRC						
Crypto						
WoR match						

Header and length parts are optional (fixed length may be used).

Individual bits may be masked in the TX header for WoR match. Configuration may match specific bit pattern or all 1s (for broadcast operation).

WoR (low duty cycle mode) can wake up at intervals and receive for a configurable period of time. If a preamble and valid sync are detected during the wakeup period then the receiver stays open until packet completion (successful or not).

Whitening polynomial: PN9 (x^9+x^5+1?)

CRC polynomials: (initial value 0x0000)

CRC-CCITT (x^16+x^12+x^5+1 = 0x1021)

CRC-16 (x^16+x^15+x^2+1 = 0x8005)

IEC-16 (0x3D65?)

Biacheva (?)

Time to enter transmit mode from shut down = 16.8 ms

Time to enter receive mode from shut down = 16.8 ms

Time to enter transmit mode from standby = 800 us

Time to enter receive mode from standby = 800 us

Time to enter transmit mode from receive = 200 us

Time to enter receive mode from transmit = 200 us

Si446x (RFM24W/RFM26W)

TBC. This has a highly configurable packet engine so should be fine.

SX1231(H) (RFM69W)

Bit order: MSB first

	Preamble	Sync	Packet Length	Address	Data	CRC
Bytes	0-65535	0-8	0-1	0-1	N	0-2
	0101.....01					
Manchester						
Whitening						
CRC						
Crypto						
WoR match						

Length and address parts are optional (fixed length may be used).

Data part is limited to 64 bytes if AES engine is used.

Suggestion to use sync as address as well (although doesn't seem to support broadcast address match). Packet handler address match (1 byte) does support broadcast detection.

WoR (Listen mode) can wake up at intervals and receive for a configurable period of time. Wakeup criteria are either a) RSSI over threshold, b) RSSI over threshold and sync detected during listen period. If the selected criterion is met than the receiver stays open until either a) packet completion, b) configurable timeout.

AES support is limited to 128-bit key. Block size if 128 bits (16 bytes)

CRC is performed on encrypted data if AES is enabled.

Whitening polynomial: PN9 (x^9+x^5+1)

CRC polynomials: (initial value 0x1D0F)

CRC-CCITT (x^16+x^12+x^5+1 = 0x1021)

The resulting CRC is complemented (XOR 0xFFFF).

CC1110 (Ciseco SRF/XRF)

Bit order: MSB first

	Preamble	Sync	Packet Length	Address	Data	CRC
Bytes	0-255	2 or 4	0-1	0-1	N	0-2
	0101.....01					
Manchester						
Whitening						
CRC						
Crypto						
WoR match						

Length and address parts are optional (fixed length may be used).

Sync word is only programmable for 2 bytes, but may be doubled up for 4 bytes (two halves must be the same).

Crypto is implemented as a co-processor feature outside of the radio.

Whitening polynomial: PN9 (x^9+x^5+1)

CRC polynomials: (initial value 0xFFFF)

CRC-16 (x^16+x^15+x^2+1 = 0x8005)

Appendix B – TinyHAN Application Protocol

This lightweight application protocol is designed for standardised transmission of sensor/control data via TinyHAN packets. It is based on binary name/value pairs and as such is extremely simple to implement on a small microcontroller. The concept of profiles is used to group data items contained within a single packet, enabling extensibility.

Format to be confirmed.

Appendix C - MQTT-SN

TinyHAN can act as a lightweight alternative to ZigBee in an MQTT-SN stack. MQTT-SN includes a number of features, namely gateway discovery and sleeping disconnects, which need not be implemented when deployed over TinyHAN. The removal of these features, not present in MQTT, allows for an easier mapping to the full-fat version of the protocol.

In a typical TinyHAN MQTT-SN network it is proposed that the TinyHAN Coordinator shall also act as MQTT-SN gateway. This obviates the need for the gateway discovery algorithm, because the coordinator address (and hence the gateway address) is already known to a registered node.

The MQTT-SN sleeping-node protocol is purely a supervision procedure, and requires the node to actively disconnect from the gateway for messages to be deferred. The node also has to perform a full connection handshake every time it wakes up. Although this is analogous to the ping-based polling approach that can be used on a non-beaconing TinyHAN network, it is a major overhead if the protocol is being deployed on a beaconing network. When deployed over TinyHAN an MQTT-SN sleeping client shall instead remain in CONNECTED state during receiver sleep periods; sleeping clients are therefore transparent to the broker. Consideration does need to be given to potentially very long round-trip times when communicating with a sleeping node, and protocol timeouts in the gateway may need to be tuned accordingly.

Running RSMB for PC-based Testing

See: http://modelbasedtesting.co.uk/?p=44

Build RSMB and create a config file containing:

```
trace_output protocol
listener 1883 INADDR_ANY mqtts
```

Run ./broker_mqtts config_file

Topic Registry

MQTT-SN supports short (2 byte) topic IDs, but this will not integrate particularly well with a full-fat MQTT server since it does not allow for wildcard subscriptions. Alternatively, full topic names may be registered at the first connection and subsequently accessed using a two byte index. Some sort of registry will be required at the embedded device to track the assigned values.

It is proposed that the lightweight MQTT-SN stack should accept topic declarations either during initialisation or using a hard-coded macro. The stack will refresh topic IDs in the registry as required by the gateway (when an inbound REGISTER packet is received), or immediately after connection (by actively REGISTERing). It shall not be possible to change or add topic declarations after the stack has been started (although IDs may change if required by the gateway). The same registry shall be used for both PUBLISH and SUBSCRIBE topics.