MyriaModem Communication Basics

MyriaNed version 2.2, document revision 6, date 2014-09-08

1. Introduction

This document will describe in more detail the MyriaNed communication implementation that is part of the MyriaModem system.

1.1. MyriaModem

The MyriaModem is a versatile wireless computing platform. It basically consists of a printed circuit board with a low energy microcontroller and a radio. The accompanying wireless firmware makes a high number of applications possible. Due to the open nature of the MyriaModem hardware, different software architectures and implementations are possible to be used.

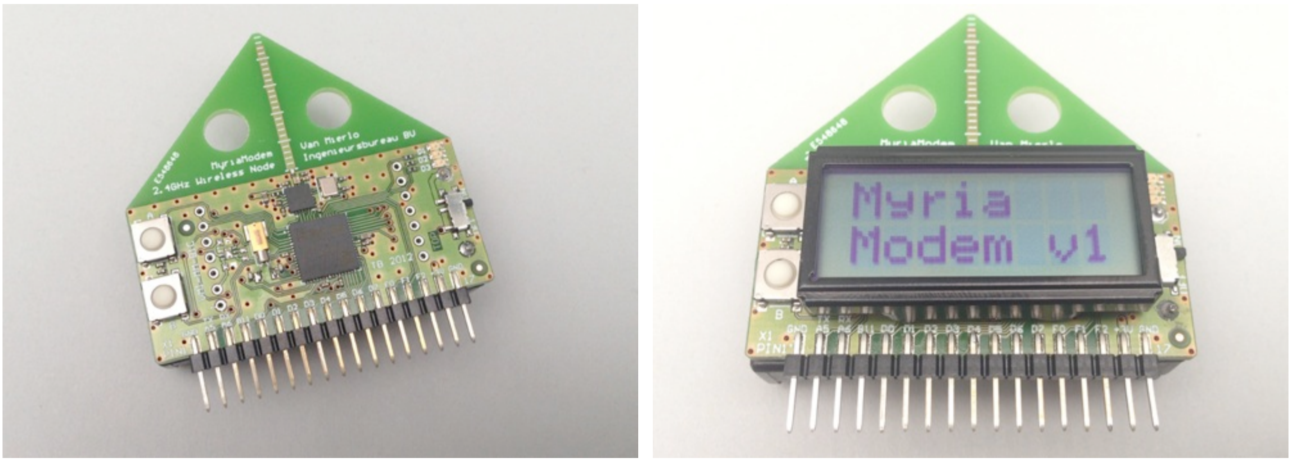


Figure 1: MyriaModem and DisplayMyriaModem hardware nodes

One of these communication software implementations is the MyriaNed™ stack. It is a highly scalable and low power network protocol. It consists of a number of software layers and conventions, that together form the basis for use with a very high number of nodes.

The development of this system started in about 2005 and is performed by DevLab in the Netherlands. In the mean time real world experiments have been deployed, with networks ranging from 10 to 1000 nodes.

1.2. MyriaNed Improvements

Due to increasing use and demands from partner developers, a small number of changes to the original software stack are proposed. The most important improvements are:

- separate main loop
- separate radio, neighbor and gossip layers
- clear timing specification
- clear packet definition
- prepared for things like encryption and network locality

The separate main loop enables the use of the MyriaModem hardware and/or MyriaNed software for other functions than only wireless sensor nodes. With full control over start up and power functions of the microcontroller, even lower energy or higher power applications are possible.

By separating the various parts of the software into layers, development of the individual parts and the complete system is easier. It becomes possible to perform experiments with only some or a mixture of the layers. It will be possible for example to create a new protocol for communicating with the neighbors, without changing the hardware or gossip application.

Better timing and packet definitions are important for interoperability. The goal of MyriaNed is to build large, multi-vendor networks. The protocol specification has to be rigid enough to make this possible, while leaving room for future innovations.

This manual further describes the resulting "MyriaNed 2" software revision 2.2 in more detail.

2. About MyriaNed

2.1. Key Features

The key features of the MyriaNed™ wireless network protocol are:

- low power
- ad hoc
- highly scalable

These features and especially the combination of all three together at the same time are unique to the Myria-Ned protocol. Other protocols of course do have parts of these features, but not all. These features make the MyriaNed protocol ideal for real world sensor networks. The start of the development for the MyriaNed protocol in 2005 was mainly fueled by the lack of these features in other protocols.

MYRIANED™
WIRELESS SENSOR NETWORK TECHNOLOGY

Figure 2: logo for MyriaNed technology

2.2. Low Power

The low power feature results in a system that consumes very little electrical energy to operate. This is achieved by putting the electronics to *sleep* most of the time. A board with radio electronics that is in deep sleep mode only consumes a couple of µA of current, while an active radio chip normally consumes tens of mA of current.

Repeatedly the microcontroller is woken from sleep, for example two times per second. One such interval is called a round. At the start of each round, the processor will start the radio communication and talk with its neighbors for a while. Typically the duration of this very short *wake* period is about 10 ms. In this way the *average* current consumption stays very low.

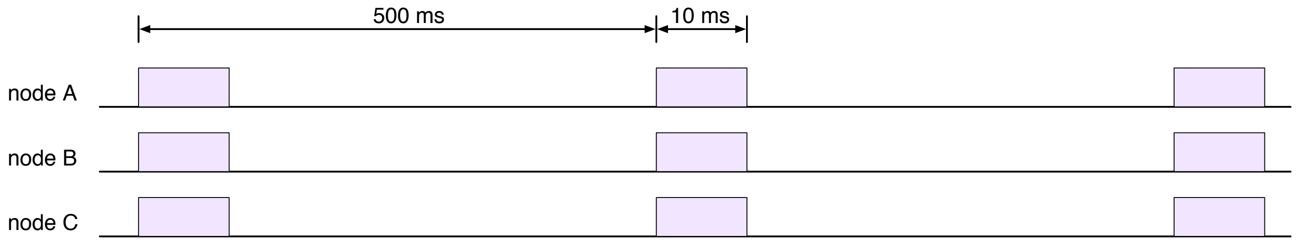


Figure 3: sleep and wake periods

If for example the *active* radio current is 20 mA, the *average* current is only 20 * 10 / 500 = 400 µA.

By using further power saving techniques, the average power consumption of the MyriaModem with 2 rounds per second is about 200 µA.

2.3. Ad Hoc

For this system with sleep and wake periods to work correctly, it is important that all the nodes in the network do this at the same time. The wake periods of all neighbors have to be synchronized, otherwise it is not possible to talk to each other.

The first step is to wait after power up until messages of other nodes are received. By careful listening this will provide timing information for the next sleep/wake rounds.

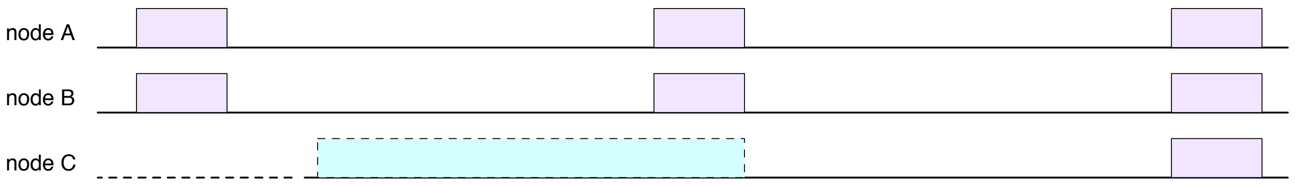


Figure 4: synchronize with the rest on startup

A drawback from this "wait on start" is that the average current consumption is a little high at power up.

This simple startup system will not work if there are a larger number of wireless nodes in the neighborhood. In this case it is possible that a split brain network will evolve. One set of nodes will be in sync together and another set of nodes will be in sync also, but not all on the same time. This way two or more separate networks will exist that have no communication in between.

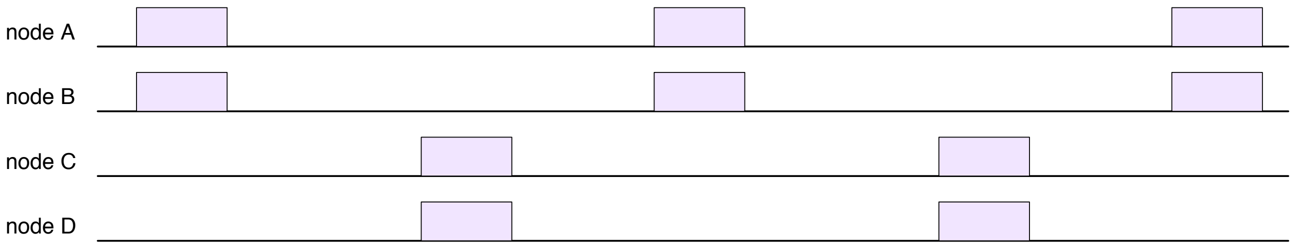


Figure 5: two completely separate networks

To accomplish the synchronization between wireless nodes in such a way that no separate networks will evolve, a *join* mechanism has been invented. With this system two groups of synchronized nodes will join together into one. The join messages are very short messages that are randomly sent while the nodes are otherwise sleeping.

In the below image the nodes A & B and C & D form two separate synchronized networks. In the sleeping period all nodes send join messages. Eventually the nodes will receive the join message from the other party and decide to align the timing. In the below figure nodes C & D react on the join message from node B.

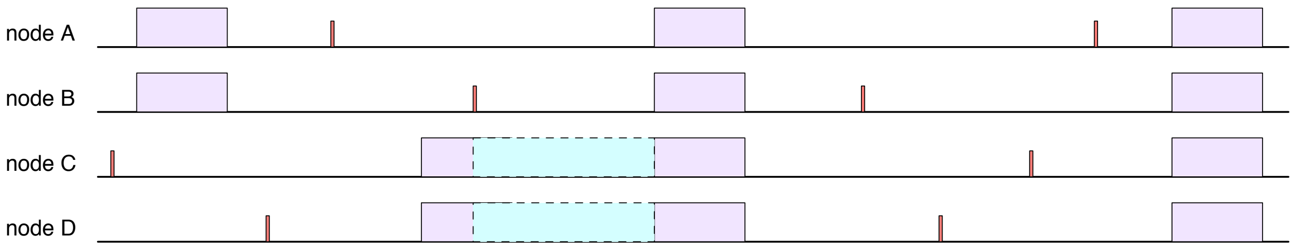


Figure 6: random join messages

By using the MyriaNed protocol, after some time all nodes will be synchronized and communicate with each other *without a central master*. By just switching on the power of a group of unrelated nodes, eventually a consistent network will start in an ad hoc way.

2.4. Highly Scalable

After the network has been synchronized with the join messages, a potentially very large group of nodes can form a network together. However each node can only communicate with nodes within reach of the small radio. For the MyriaModem radio this is about 20 meter. It is for the electronics not possible to exchange data with nodes that are farther away.

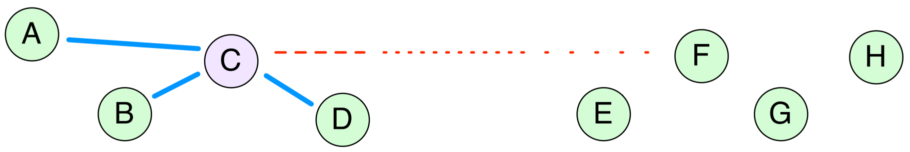


Figure 7: talk with neighbors only

Node C in the above figure can reach A, B & D but not any of the other nodes.

On the lowest protocol level, the nodes therefore only talk with the direct neighbor nodes in their vicinity. The direct communication is only in a small area. To get communication farther away through the network, all the nodes have to pass messages through to each other. For this behaviour a mechanism called *gossip* is used.

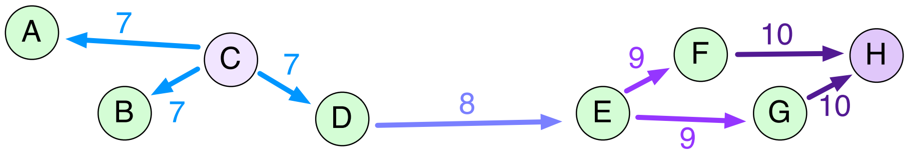


Figure 8: pass messages through

In the example figure above, node C has a message. In round 7 it gets passed to the neighbor nodes A, B and D. In round 8 node D will pass the message to its neighbor E. In round 9 and 10 the message is told to nodes F, G and eventually node H. Although node C and H can not reach each other directly, the message is transferred through the network in a small number of rounds.

Gossip works with *news items*. Only new messages will be told to neighbors. A message is only new for a certain amount of time, after which it gets outdated and is not considered news any more. In the gossip protocol a system called *history* is used to ensure that messages stop traveling through the network.

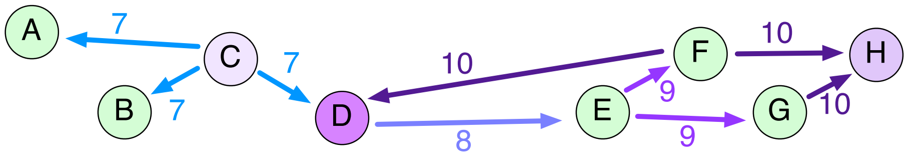


Figure 9: stop old news

In the above figure the message gets back at node D in round 10. The message has the exact same history number as when it left node D in round 8. Node D will compare the history number with the one in its *cache* memory and will decide that the message is old news. Depending on the network settings, node D will discard the message and it will not retransmit this message to its neighbors.

2.5. Constraints

The gossip system will reliably work over thousands of nodes as long as the number of newly sent messages is relatively low.

For developing a MyriaNed protocol application the following basic constraints apply:

- the messages are small, about 20 bytes maximum
- only one message is transmitted each round
- the network throughput is therefore relatively low
- about 10% of the packets that are transmitted get lost due to collisions
- for the message pass-through to work, the number of new messages per round is limited
- do not overfeed the gossip layer with too many new messages per minute

If there are more new messages than the gossip layer can handle, the network will still work but the new messages will just not travel to all the edges. In that case some nodes will more or less get isolated in their neighborhood. For most decentralized sensor applications this is not a restriction.

A general rule of thumb for good throughput is about one new message for the same number of rounds as there are nodes in the network. Example: 1 new message each 100 seconds in a 100 node network with 1 second round time.

With the newer *time to live* functionality, part of this burden is relieved since it will keep "local" messages from clogging the network. The *time to live* value is expressed in a number of rounds. An infinite *time to live* is possible using a specific value.

Another important constraint in practice, is that the network is used on radio frequencies that are open to all people. Everybody is able to listen to all communication. Everybody is therefore also able to inject new messages or otherwise disturb the communication.

The newer *encryption* functionality will be a solution for part of this problem. By encrypting messages it is possible to shield listeners from the transmitted information. Moreover, rogue senders are not able to participate in the gossip system and therefore won't disturb the rest of the network anymore.

3. MyriaNed Definition

3.1. Layers

The MyriaNed protocol is divided in a number of layers, that each perform a specific task or interface:

■ radio	interaction with the communication hardware
■ neighbor	communication with *in-range* neighbors
■ gossip	process messages and let them travel through the network
■ application	the program performing a networking function
■ main	the scheduler and interface to the other layers and node hardware

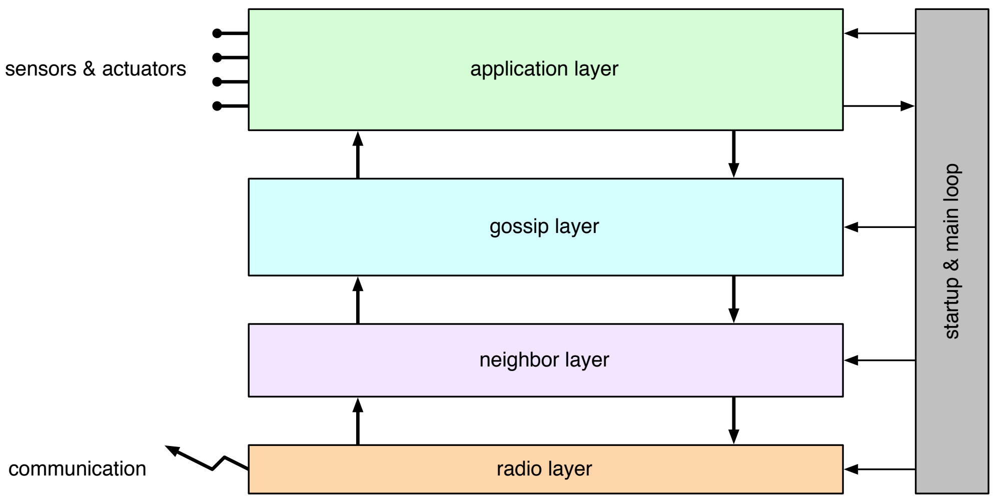


Figure 10: protocol layers and their relation

By separating the functions in more or less independent layers, the development, implementation and use of the protocol is highly simplified. The protocol developer is able to replace a layer implementation with another, without changing the higher or lower layers it has interaction with.

3.2. Timing

In the world of wireless networking, different physical radios give incompatible communications. For a network with the Nordic Semiconductor chip `nRF24L01P` the timing definition for MyriaNed is detailed in this chapter. If one wants to use MyriaNed with another radio interface, most probably other timing will be used.

3.2.1. Key Elements

The most important timing parts and terms of the MyriaNed protocol are:

■ packet	fixed structure of bits that is transmitted
■ slot	the smallest part of the transmissions
■ block	a number of slots for scheduling transmissions
■ round	repeat period for transmission and sleeping
■ join	message to invite other nodes to join my network

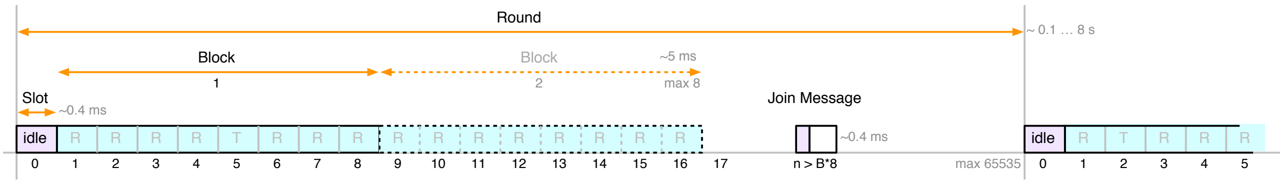


Figure 11: timing definition

3.2.2. Packet

For the `nRF24L01P` radio chip the payload of a physical over the air packet is always 32 bytes. Before and after the payload a number of bits are used for radio synchronization and transfer verification. All communication of the higher MyriaNed layers is performed in chunks of this 32 byte payload. The transmission of a packet over the air takes about 170 µs.

3.2.3. Slot

Due to oscillator startup and drift times, receive and transmit switchover times and related things, the minimum time for MyriaNed action is about 0.4 ms. This is called a slot. All actions on the radio side of the protocol are defined as a number of slots.

The slot time is exactly (1 s / 32768) * 14 = 427 µs.

The specific timing within a slot is not detailed in this document.

3.2.4. Round

The transmission of packets to neighbors is scheduled in rounds. This is sometimes also called a frame. A typical round time is for example 500 ms. The system is capable of rounds up to 32767 slots, about 14 seconds. The transmission of packets with neighbors is normally at the start of a round, after which the node may go to sleep if it wishes.

For practical reasons the round time is often limited to a factor of 2, with possible values being 125 ms, 250 ms, 500 ms, 1 s, 2 s, 4s, and 8 s.

3.2.5. Block

Communication is performed in strings of 8 slots. Depending on the number of neighbors that are in reach, MyriaNed can use more blocks for the communication. Each node sends its message once per round inside one block. The block and slot position is randomly chosen. There is a maximum of 8 blocks.

Due to the random nature of the slot selection, it is highly likely that multiple neighbors will transmit at the same time. Therefore packet loss due to collisions is inevitable. A solution is to resend new messages more often, so to increase the chance that eventually the message is transferred.

But the system can also decide to use more blocks for transmission. This way the chance that packets get through to other neighbors is higher because there is more time for collision free transmissions.

3.2.6. Join

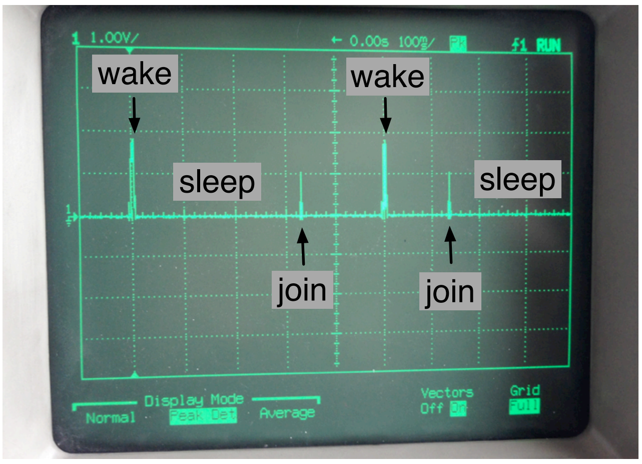
The join messages are transmitted randomly in the empty tail of a frame. In the message at least the actual slot number is transmitted. This message is an invitation for other nodes to join the timing of this network. If there happen to be wireless nodes with the same protocol but other timing, they are now able to match their frame timing.

During normal operation a MyriaNed network will not see each other’s join messages, they only listen during one of the initial transmission blocks. But freshly started nodes or nodes from another network may be able to capture them.

A join message can operate without any data contents. However, to be able to use the over-the-air bootloader functionality extra information is included. At least the node shall supply its unique Node ID and preferable also some information about the current active firmware or application.

3.2.7. Example Measurement

With a current monitor on the power supply of a MyriaModem it is possible to give detailed insight into the protocol. This way the different parts of the communication are visualized in an indirect manner.



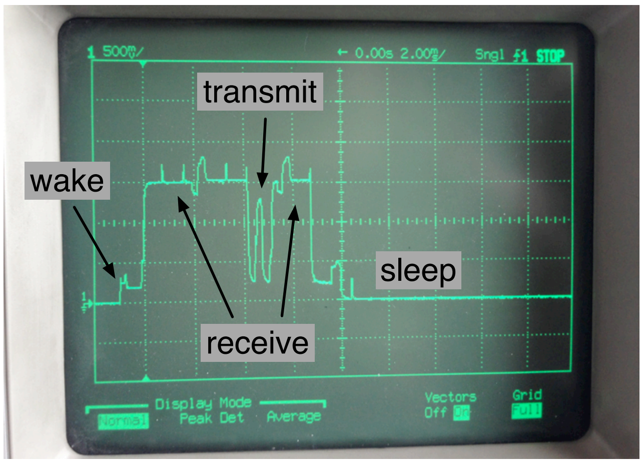


Figure 12: current monitor results on oscilloscope

In the left picture one horizontal division is 100 ms and one vertical division is 10 mA. Each 500 ms a new round starts with the high current level of an active processor and radio. At random somewhere in the round a join message is visible. In between the wake and join parts the node is sleeping with almost zero current.

In the right picture one horizontal division is 2 ms and one vertical division is 5 mA. About a millisecond before the receiver is active the processor wakes up and starts using a bit of current. The active receiver uses the most power in the MyriaModem. At slot 6 in the communication block, one slot is used for transmission and the other slots are used for reception. In the MyriaModem the transmitter uses less power than the receiver, even at the 0 dBM power level.

After the whole block of receiving and transmitting has passed, the processor still has some processing to do before going to sleep again. Sleep current is about 1 μA if no other peripherals than the 32 kHz timer stay active during deep sleep. In this older measurement the wake time is about 9 ms, with the newer libraries the wake time is only about half of this.

In the MyriaModem Edukit a Current Monitor board is supplied. This will give you the possibility to make these kind of measurements with this level of detail by yourself. Especially for debugging energy consumption or synchronization aspects of the protocol this is a very valuable tool.

3.3. Packets and Layers

The 32 byte wireless payload of the `nRF24L01P` radio chip are the starting point for the packet definitions of the MyriaNed layers. If one wants to use MyriaNed with another radio interface, most probably other packet definitions will be used.

3.3.1. Overview

Each layer has its own subpart of a packet. The radio chip uses the native over-the-air wireless packet. The radio layer performs validity checks and in the future also packet encryption. The neighbor layer adds timing to these packets. The gossip layer adds validation to the payload of the application layer.

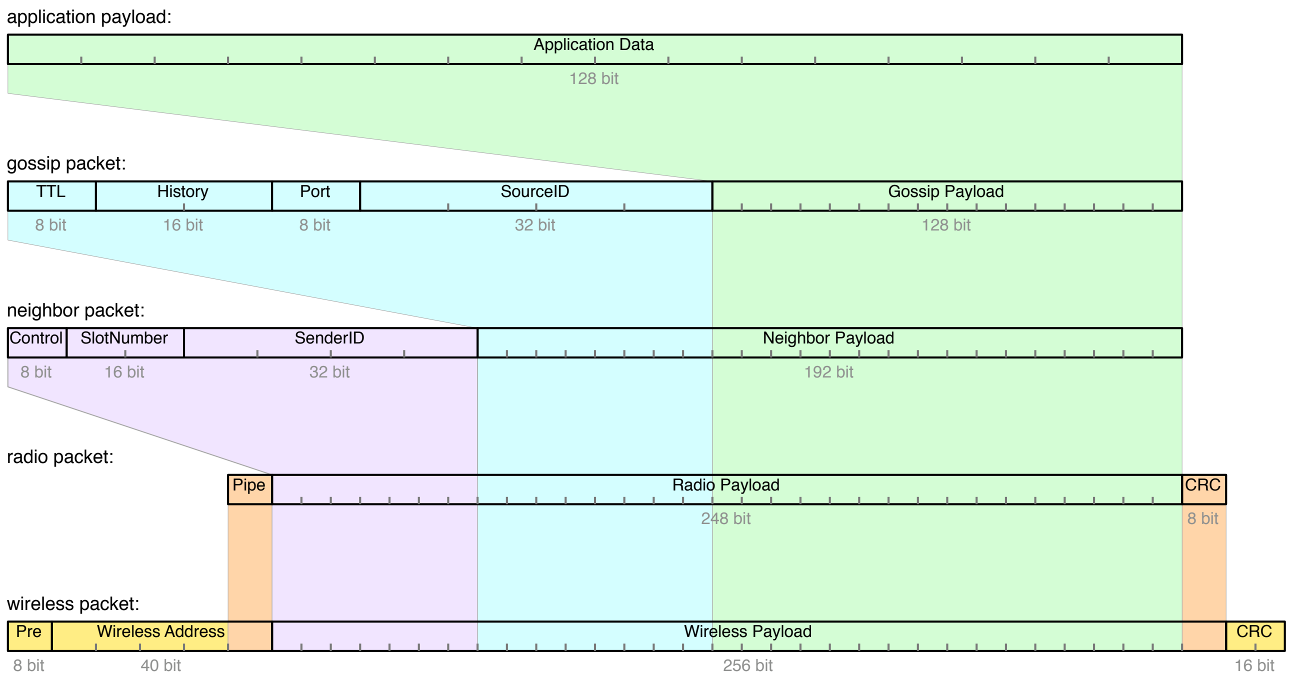


Figure 13: packet definition (drawing not to scale)

3.3.2. Wireless Layer

The wireless layer transfers packets with a payload of 32 bytes *over-the-air*. The packet consists of 8 bit preamble, 40 bit wireless address, 256 bit payload and 16 bit CRC.

Packets with CRC errors are discarded. Transmissions on another RF channel are not seen by neighbors. Transmissions with different Wireless Addresses are not seen by neighbors. By changing the RF channel or the Wireless Address it is possible to run multiple networks more or less alongside each other.

The default Wireless Address for unencrypted MyriaNed messages is `0x4D 0x79 0x72 0x69 0x61`, which translates to the string "`Myria`".

3.3.3. Radio Layer

Depending on the radio chip functionality, the radio layer can use multiple Wireless Address pipes. For example one Wireless Address can be used for the bootloader. Another is to separate encrypted messages from regular messages.

At the end of the radio layer packet 8 extra CRC bits have been added to facilitate to check if an encryption key is correct.

The radio layer of MyriaNed 2.2 implements three possible pipes with different wireless addresses:

- regular messages: passed to the neighbor layer
- encrypted messages: check key and decrypt before further processing
- bootloader messages: transparently reset the node for wireless reprogramming

In the programming API the pipe number a message was received on is available.

3.3.4. Neighbor Layer

The neighbor layer starts with an 8 bit control field for the type of message and other details that are necessary for managing the communication. A message can be a regular message or a join message.

The next 16 bits contain the slot number, that is used to share precise timing information between nodes. The actual slot time is used to adjust various timings between nodes to keep the network in sync.

The next 32 bit is the node ID of the sending or retransmitting node. This unique number is used to distinguish between nodes for network management tasks and for example the bootloading process.

The remaining 192 bit message payload it receives from its neighbors are passed to the higher layers.

If available from the upper layer, each period it will transmit a packet of its own. When no transmit packet is available, the neighbor layer will still transmit on the radio an empty packet consisting of a payload filled with zeros. This is to keep the network synchronized even when no higher level data is transferred.

The neighbor layer will also always transmit a join message during each round.

3.4. Gossip Layer

The gossip layer adds the following fields:

- 8 bit TTL
- 16 bit history
- 8 bit port
- 32 bit source ID

With this information it will perform various functions:

- packets that are over the time to live can be discarded without application intervention
- packets from different applications or application functions can have different port values
- a caching algorithm can use the port, source and history fields
- depending on the caching algorithm used, the 20 bytes payload is passed to the application
- based on the cache it will select a packet to transmit by the neighbor layer in the next round

Different applications may require different cache algorithms. It is possible to adapt the caching and packet selection based on the port field. Currently there are two main cache algorithms implemented, each with a different function:

- local data function: the application generates data which is specific for its own node, for example a temperature sensor
- global data function: the application generates data which shall be common for all nodes, for example a global time or state

For more information about the use of the gossip layer, see the descriptions and examples in the library documentation.