LoRaWAN Network Server Demonstration: Version 2.1.1 Release note

1 History

Revision	Modification / Remarks / Motive	Author
1.0	Document created	AMa

2 Release Features

This is the release of the LoRaWAN IoT Demonstration Network Software Solution. The intent of this software package is to allow the user to easily create an entire system for connecting a LoRaWAN gateway and allowing a user to access the data and control the remote devices. This release includes the following items:

1. Server Description
2. Installation Guide
3. Server Build Guide
4. Gateway to Server Interface Definition
5. Inter-Server Interface Definition
6. Configuration Command Description
7. Demonstration Network Software Solution source code file set.

The source code provides the following functionality:

1. Network server
2. Application server
3. Customer server
4. Network controller (including ADR)

3 System Requirements

Hardware Requirements

No specific hardware requirements. The hardware platform should be selected with the size and performance requirements of the overall system in mind.

Operating System Requirements

Windows 7.0
Microsoft Visual Studio 2010
MySQL version 5.6.16

Linux flavour and version: CentOS v6.4
Based on x86_64-redhat-linux
Compiler version: gcc version 4.4.7 20120313 (Red Hat 4.4.7-3) (GCC)
MySQL version: 5.1.69

Raspberry Pi Hardware version BCM2708, Revision 0010
OS version: Debian v7.6
Linux version: 3.12.31
Compiler version: gcc (Raspbian 4.8.2-21~rpi3rpi1) 4.8.2
MySQL version: 5.5.40-0+wheezy1

4 Enhancements

4.1 MYSQL error message

Previously the MYSQL error status was always written as '1' to the log file. This has now been modified so that actual MYSQL error is written.

4.2 LoRa server now automatically updates the MYSQL database

Previously users would have to manually import a database schema to MYSQL if the tables had changed. Now the LoRa server will automatically update the MYSQL tables if the format has changed. The database user will need to have CREATE, DROP, INDEX and ALTER privileges to do this.

4.3 Ability to set region for server and gateway

Motes in different regions will transmit on different frequencies – e.g. Europe 863 format transmits between 863Mhz to 870Mhz. The LoRa network server now handles this for Europe 863, USA 902, China 779 and Europe 433 regions. The network server will reject messages from a mote if they are transmitted on an incorrect frequency for that region. The network server can either be configured for a particular region or it can be configured to handle gateways in different regions.

4.4 Gateway delete

It is now possible to delete gateways from the configuration command 'gateway delete <eui>'

4.5 Gateway position

The network server transmits the position of each gateway to other LoRa servers receiving the "gwst" application server every 60s even when the position has not changed.

4.6 Ability to add mote to named app

In the Customer Server it is now possible to use a configuration command to add a mote to a named application instead of by EUI.

4.7 Setting Tx power too high

Setting the Tx power of the gateway too high using the network server commands will now result in an error message.

4.8 Second window data rates

The data rate used for second window transmissions is now configurable for each region.

4.9 Downlink application data queue added to the Customer Server

A downlink application queue has now been added to the customer server. This is used for queuing messages to be sent to the mote.

4.10 Provisioned mote network address

It is now possible to have a provisioned mote's network addresses unrelated to its EUI.

4.11 EU863 Frequency Band has been widened

The EU frequency band has now been widened in the network server to have a minimum value of 863,000,000 and a maximum value of 870,000,000

5 Fixes

5.1 Zero count of upstream frames from gateway

Previously if the count of upstream frames as sent by the gateway to the LoRa network server was zero then this would cause an error when writing to the MYSQL database.

5.2 Long integers now correctly handled

Long integers were not handled correctly as they would be a different size on 64 bit Ubuntu than the Windows and 32 bit Ubuntu versions.

5.3 Gateway position now read from the database

The value of gateway attribute allowGpsToSetPosition was not read from the network server database.

5.4 Gateway created with incorrect syntax now have default values

When a gateway is created with an incorrect configuration command parameter it is now configured to have the default values.

5.5 Network Controller deadlock

Previously there was a mutex deadlock in the network controller when calling GetPositionStatus.

5.6 Null application EUI was incorrect

Previously the network server null application EUI was set to Application::invalidEui. It is now set to Application::nullEui. This would happen when sending the gateway status to clients.

5.7 Network Server transmit frames sent with incorrect 'tmst' values

The network server would send frames to a mote with incorrect 'tmst' values. The transmit frames were being sent with '5,000,000' 'tmst' value instead of the correct '5,000,000 plus the Receive Time'.

5.8 Applications with MS Bit in EUI Set

Applications with the MS Bit set in the EUI would be stored incorrectly in the database due to conversion between unsigned to signed.

5.9 Frame header options

Previously a zero length mote Frame Header Options field would be forwarded by the network server.

5.10 Frequency rounding

Frequencies parsed in Gateway Messages received on the network server would be incorrectly rounded up.

5.11 Network Server log file

The network server log file incorrectly wrote the title 'Network Address' instead of the actual 'App EUI'. This has now been fixed.

5.12 Ghost frame incorrect frequency

When a mote is very close to a gateway a frame can be received on an incorrect adjacent channel as well as the correct wanted channel. This can result in the network server receiving 'ghost frames' in addition to receiving the correct frame.
Previously when this occurred and the received 'ghost frame' was sent from a gateway to the network server before the genuine frame had arrived, the incorrect frequency of the adjacent channel frame was sent to the application server instead of the genuine wanted frequency.

5.13 Join Accept Frame received from Gateway was reported as 'too short'

Sometimes a gateway can receive join accept messages that it has transmitted and forward these back to the network server. In this case the network server would report the frame as being 'too short' – now it will discard the frame.

5.14 Network Server would take up all the cpu capacity

Previously the network server time thread would take up all the pc cpu capacity due to the snooze call not being set correctly.

5.15 Incorrect Spreading Factor Used in USA Downlink Messages

When the USA end device transmits with data rate DR4 (SF8BW500) the server should respond on first reception window with DR13 (SF7BW500) but it was responding with DR12 (SF8BW500).

5.16 Downstream frame from Network Server maybe corrupted

If a downlink MAC command and a downlink application data packet together exceed the maximum frame size, the frame may have been corrupted.

5.17 Gateways Read from Database with Wrong Region

In release 2.1.0 gateways read from the database were set with an unknown region when they were entered in a previous database version.

6 Known Issues and Problems

6.1 Default downlink 2nd window spreading factor

The network server default downlink 2nd window spreading factor in the European 863MHz and 433MHz bands is SF9, while the LoRaWAN specification [1] default value is SF12. This value may be changed by a network server configuration command.

7 Compliance

LoRaWAN 1.0 Section	Requirement	R2.1.x Status
1.2	Data transmitted as little endian	Compliant
2.1	Be able to handle Class A end device	Compliant
2.1	Be able to handle Class B end device	Not Implemented
2.1	Be able to handle Class C end device	Not Implemented
3.1	Uplink messages sent by end devices - physical header (PHDR) and header PHDRCRC are included	Compliant
3.2	Downlink messages sent by NS to only one gateway	Compliant
3.2	Downlink messages contain PHDR and PHDR_CRC	Compliant
3.3.1	NS able to transmit to end device during first receive window	Compliant
3.3.2	NS able to transmit to end device during second receive window	Compliant
4	Handle uplink messages with PHY layer, PHYPayload, MACpayload, FHDR and MIC	Compliant
	Handle downlink messages with PHY layer, PHYPayload, MACpayload, FHDR and MIC	Compliant
4.1	PHYpayload - contains MHDR, MACPayload and MIC	Compliant
4.2	MHDR contains Mtype, RFU and Major version	Compliant
4.2.1	Handle Join Request, Join Accept, Unconfirmed Data Up, Unconfirmed Data Down, Confirmed Data up, Confirmed Data Down, RFU and Proprietary	Compliant[1]
4.2.1.2	Confirmed data has to be acknowledged	Compliant
4.2.1.2	Unconfirmed data does not require an acknowledgement	Compliant
4.2.2	Major version of data message should be set	Compliant
4.3	MACPayload contains FHDR, Fport and FRMPayload	Compliant
4.3.1	FHDR contains DevAddr, FCtrl, FCnt, Fopts	Compliant
4.3.1	FCtrl contains ADR, ADRACKReq, ACK, Fpending, FOptsLen	Compliant
4.3.1.1	ADR, ADRACKReq - if ADR bit is set the network will control Data Rate through MAC Commands.	Compliant
4.3.1.1	ADR bit may be set/unset on demand and the network should handle this	Compliant
4.3.1.1	If ADRACKReq bit is set the network shall respond with a downlink frame set by the ADR_ACK_DELAY	Compliant
4.3.1.1	If the network hasn't responded to the above then it should handle the end device transmitting on the next lower data rate.	Compliant
4.3.1.2	If confirmed data up is received the network will send an acknowledgment using one of the receive windows opened by the end device	Compliant
4.3.1.4	FPending is only used in downlink indicating the gateway has more data pending to be sent so ask the end device to open another receive window	Not Implemented
4.3.1.5	The NS tracks the uplink FCntUp and generates the downlink counter FCntDown.	Compliant
4.3.1.5	After join accept frame counters for that end device are set to 0	Compliant

[1] There are no proprietary messages used

LoRaWAN 1.0 Section	Requirement	R2.1.x Status
4.3.1.5	If too many frames missing ie the difference in frame count is greater than MAX_FCNT_GAP then too many frames have been lost so subsequent ones are discarded.	Compliant
4.3.1.6	NS needs to know width of frame counter used by end device ie 16 or 32 bits - this is done with the FOptsLen in the FCtrl byte	Partially Compliant[2]
4.3.1.6	Fopts transport MAC commands of up to 15 octets that are at the end of data frames	Compliant
4.3.1.6	MAC commands cannot be simultaneously in in payload field and frame options field	Partially Compliant[3]
4.3.2	Fport - if frame payload field is not empty the port field must be present. Fport value of 0 indicates that FRMPayload contains MAC commands only.	Compliant
4.3.2	FRMPayload length should be <= M-1-FHDR len. (where M is the maximum MAC payload length)	Compliant
4.3.3	FRMPayload must be encrypted before the MIC is calculated	Compliant
4.3.1.1	Encryption - the key used depends on the FPort of the data message and so will be either the nwkSKey or the AppSKey	Compliant
4.3.1.1	The fields encrypted are the FRMPPayload	Compliant
4.3.3.2	If layers above LoRaWAN provide encrypted FRMPayload then LoRaWAN transfers FRMPayload from MACPayload to the application and FRMPqayload from the application to MACPayload without anny modification of FRMPayload.	Compliant
4.4	MIC calculated over MHDR, FHDR, FPORT, FRMPayload	Compliant
5	MAC Commands not visible to the application of application server or application running on the end device.	Compliant
5	MAC commands are piggybacked in the Fopts field or in the FRMPayload field with the Fport field set to 0.	Compliant
5	Piggybacked MAC commands are always sent without encryption and must not exceed 15 octets	Compliant
5	Mac commands sent as FRMPayload are always encrypted and must not exceed FRMPayload length	Compliant
5	MAC commands go from CID =0x02 to CID=0x08	Compliant
5	Length of MAC command is not explicitly given and must be implicitly known by the MAC implementation	Compliant
5	MAC commands canot be skipped so the first unknown MAC commands terminates the processing so it is advisable to order them according to the version of LoRaWAN that has implemented them for the first time.	Compliant
5.1	LinkCheckReq responded to with LinkCheckAns	Compliant
5.2	LinkADRReq can be sent by NS and handles LinkADRAns response	Partially Compliant[4]
5.2	LinkADRReq contains ChMask data control channels	Compliant

[2] 16bit Frame Counter not supported
[3] It is possible to instruct the Network Server to send MAC commands in both Header and Payload of the same message
[4] Response ignored

LoRaWAN 1.0 Section	Requirement	R2.1.x Status
5.2	LinkADRReq contains Redundancy bits to control NbRep for uplink messages and ChMaskCntl to further control the ChMask	Compliant
5.2	NS handles LinkADRAns values in data rate Ack	Not Implemented
5.2	NS handles LinkADRAns values in Channel mask Ack	Not Implemented
5.2	NS handles LinkADRAns Power ACK values	Not Implemented
5.3	DutyCycleReq sednt by NS to limit max agg transmit dudty cycle. Handles DutyCycleAns	Not Implemented
5.4	RXParamSetupReq to change freq and data rate for 2nd recv window by setting DR offset and rx2 data rate and the frequency. Handle response of RxParamSetupAns with each of the various acks in it	Compliant[5]
5.5	DevStatusReq can be sent and handle DevStatusAns	Not Implemented
5.6	NewChannelReq to modifyt channel parameters such as freq and data rates. Should handle newChannelAns	Not Implemented
5.7	RxTimingSetupReq to configure delay between the end of the tx uplink and the first reception slot. Handle RxTimingSetupAns	Not Implemented
6.2.4	Join Request message contains appeui, deveui and devNonce and should ignore requests with previous devNonce.	Compliant
6.2.5	Join Accept should be sent by the NS and uses JOIN ACCEPT DELAY 1 AND JOIN ACCEPT DELAY 2	Compliant[6]
6.2.5	The channel freq and rate for the two windows are idential to rx1 and rx2 receive windows.	Compliant
6.2.5	Join accept should contain appNonce, Netid, DevAddr, DLSettings, RxDelay and an optional list of channel frequencies.	Compliant[7]
6.2.5	MIC value calculated for join accept	Compliant
6.2.5	Join accept should be encrypted with AppKey	Compliant
6.2.5	NetId should be calculated with 7 MSB of the short address fo an end device.	Compliant
6.2.5	DL settings contain the downlink configuration. - the rx1 DR offset and the RX2 data rate.	Compliant[8]
6.3	Activation by Personalization	Compliant
6.3	nwkSKey and AppSkey are stored in end device instead of DevEui, Appeui and Appkey.	Compliant
7	WorldWide Settings	
7.1	EU 863	
7.1.1	Use preamble format	Compliant
7.1.2	Channel frequencies to be used	Compliant
7.1.2	Duty cycle limit to be followed	Not Implemented
7.1.2	Data rates to be followed	Compliant
7.1.2	863 to 870Mhz for end device	Compliant
7.1.3	Data rate and end point output power encoding	Compliant
7.1.4	Join Accept channel list	Compliant

[5] This message is never sent
[6] Never uses 2nd window, no gateway scheduling in this version
[7] Channel freq not sent, RX1DRoffset fixed at zero, RX2 Data rate fixed at 3, RxDelay fixed at 0 (1 sec)
[8] RX1DRoffset fixed at zero, RX2 Data rate fixed at 3

LoRaWAN 1.0 Section	Requirement	R2.1.x Status
7.1.5	LinkAdrReq - chMask to be followed	Compliant
7.1.6	Max payload size	Compliant
7.1.7	Receive window data rates	Compliant
7.1.8	Default settings for delays etc	Compliant
7.2	US902	
7.2.1	Use preamble format	
7.2.2	Channel frequencies to be used	Compliant
7.2.2	Duty cycle limitation to be followed	Not Implemented
7.2.2	Data rates to be followed	Compliant
7.2.2	902 to 928Mhz for end device	Compliant
7.2.3	Data rate and end point output power encoding	Compliant
7.2.4	Join Resp CF list not supported	Compliant
7.2.5	LinkAdrReq - chMask to be followed	Compliant
7.2.6	Max payload size	Compliant
7.2.7	Receive window data rates	Compliant
7.2.8	Default settings for delays etc	Compliant
7.3	CN779	
7.3.1	Use preamble format	
7.3.2	Channel frequencies to be used	Compliant
7.3.2	Duty cycle limitation to be followed	Not Implemented
7.3.2	Data rates to be followed	Compliant
7.3.2	779 to 787Mhz for end device	Compliant
7.3.3	Data rate and end point output power encoding	Compliant
7.3.4	Join Accept CF list	Compliant
7.3.5	LinkAdrReq - chMask to be followed	Compliant
7.3.6	Max payload size	Compliant
7.3.7	Receive window data rates	Compliant
7.3.8	Default settings for delays etc	Compliant
7.4	EU433	
7.4.1	Use preamble format	
7.4.2	Channel frequencies to be used	Compliant
7.4.2	Duty cycle limitation to be followed	Not Implemented
7.4.2	Data rates to be followed	Compliant
7.4.2	433.175 to 434.665Mhz for end device	Compliant
7.4.3	Data rate and end point output power encoding	Compliant
7.4.4	Join Accept CF list	Compliant
7.4.5	LinkAdrReq - chMask to be followed	Compliant
7.4.6	Max payload size	Compliant
7.4.7	Receive window data rates	Compliant
7.4.8	Default settings for delays etc	Compliant

8 References

[1] LoRa Alliance, "LoRaWAN Specification," LoRa Alliance, 2015.