LoRaWAN Network Server Demonstration: Implementation Description

1 History

Revision	Modification / Remarks / Motive	Author
1.0	Document created	DRo

2 Introduction

This document adds to the server description document [1] providing more internal detail of the operation and interfaces of the LoRa™ servers (including the network controller).

The LoRa network server (NS), application server (AS) and network controller (NC) are licensed as part of the Semtech 'LoRa IoT Reference Network Software Solution'.

The LoRa customer server (CS) is licensed in the same way. The CS simply receives data from the AS and either stores it in a relational database or appends it to a text file. It is expected that the CS will be largely replaced or completely replaced in any operational LoRa system.

3 Threads

Each server, with the exception of the NS, uses an identical top level ("C" main()) function, which starts the following threads:

Time Thread:	Runs in response to the passage of time. Performs the time related operations that the server requires.
UDP receive thread:	Waits for messages to be received from either the UDP port (or ports, in the NS). In the NS, the thread logs the time of receipt and passes the message to the inter-thread queue. In the remaining servers, the thread performs the operations required by receipt of the message.
TCP wait thread:	Waits for an incoming TCP connection request and then opens the requested connection.
TCP connect thread:	Runs in response to the passage of time. Attempts to connect to remote servers in order to establish the TCP connections required by the current configuration.
TCP receive thread:	Waits for data to be received from an existing TCP connection.

The NS has 2 further threads. These threads improve the network server's performance.

UDP service thread:	Waits for UDP messages previously received by the 'UDP receive thread' to be available in the 'inter-thread queue' and performs the operations required by receipt of the message.
Gateway transmit thread:	Runs in response to the passage of time. The sole function of the thread is to transmit time-critical messages to the gateways.

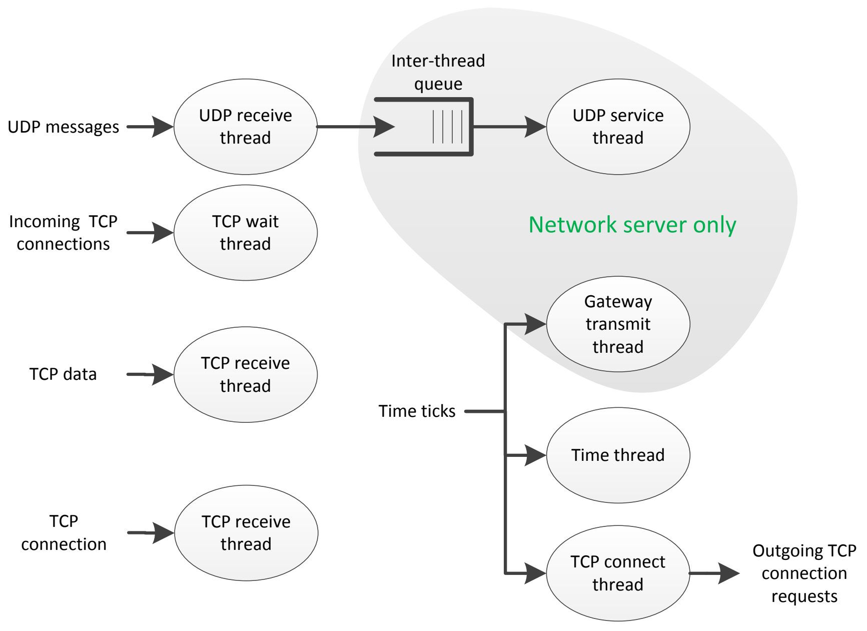
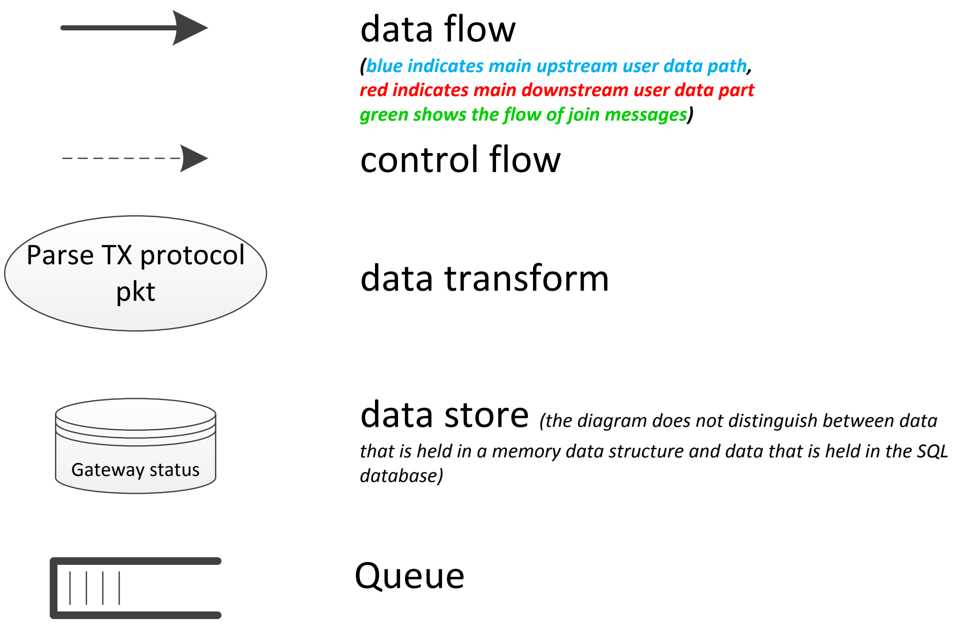


Figure 1: Thread diagram (of a single server)

4 Data flow

4.1 Diagrams



data flow
(*blue indicates main upstream user data path*, *red indicates main downstream user data part* *green shows the flow of join messages*)

control flow

data transform

data store *(the diagram does not distinguish between data that is held in a memory data structure and data that is held in the SQL database)*

Queue

Figure 2: Key to dataflow diagrams

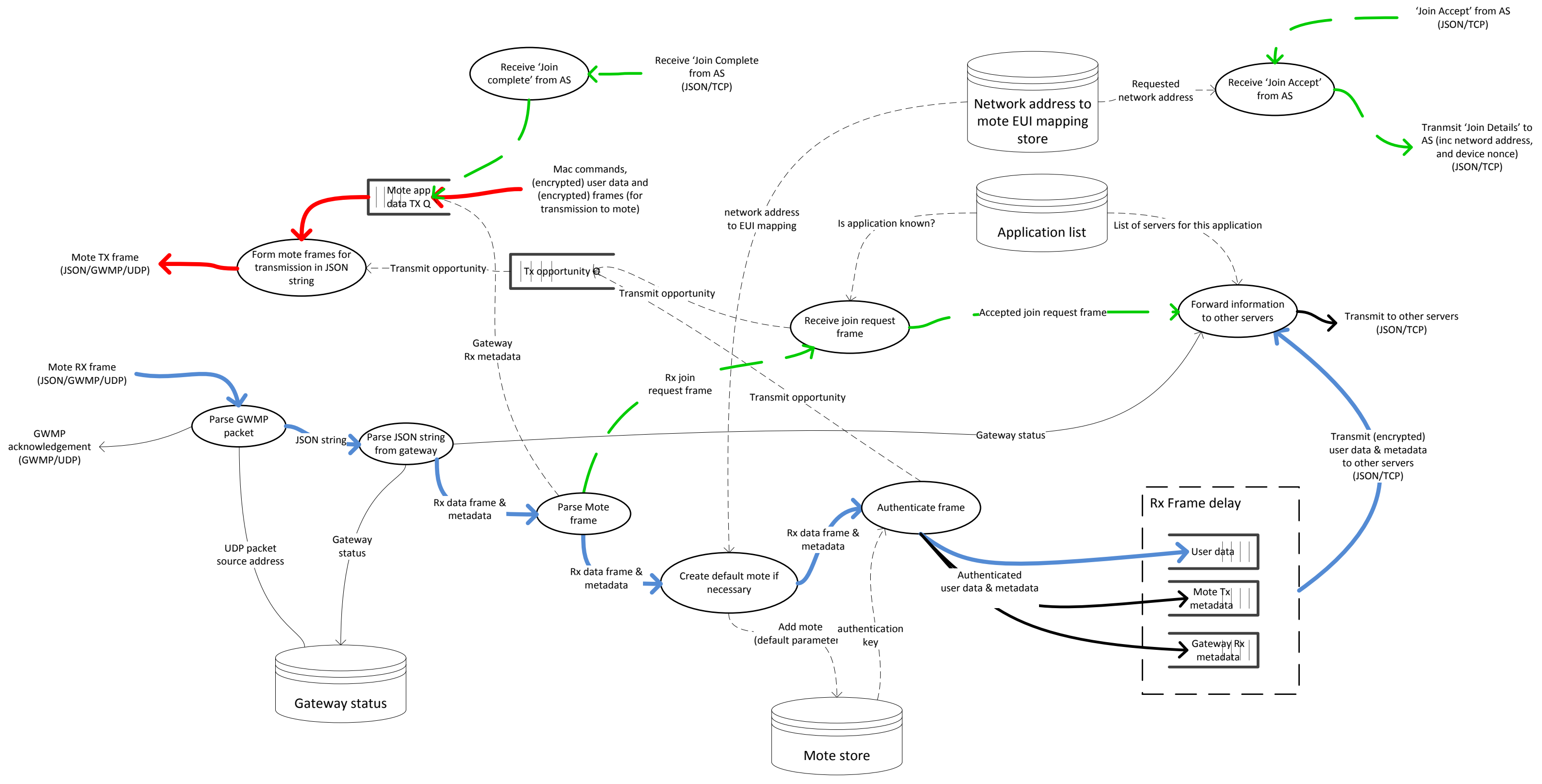

'Join Accept' from AS (JSON/TCP)
Receive 'Join Complete from AS (JSON/TCP)
Receive 'Join complete' from AS
Network address to mote EUI mapping store
Requested network address
Receive 'Join Accept' from AS
Tranmsit 'Join Details' to AS (inc netword address, and device nonce) (JSON/TCP)
Mote app data TX Q
Mac commands, (encrypted) user data and (encrypted) frames (for transmission to mote)
network address to EUI mapping
Is application known?
Application list
List of servers for this application
Mote TX frame (JSON/GWMP/UDP)
Form mote frames for transmission in JSON string
Transmit opportunity
Tx opportunity Q
Transmit opportunity
Receive join request frame
Accepted join request frame
Forward information to other servers
Transmit to other servers (JSON/TCP)
Gateway Rx metadata
Mote RX frame (JSON/GWMP/UDP)
Rx join request frame
Transmit opportunity
Parse GWMP packet
GWMP acknowledgement (GWMP/UDP)
JSON string
Parse JSON string from gateway
Gateway status
Transmit (encrypted) user data & metadata to other servers (JSON/TCP)
Rx data frame & metadata
Parse Mote frame
Authenticate frame
Rx Frame delay
Rx data frame & metadata
User data
UDP packet source address
Gateway status
Rx data frame & metadata
Create default mote if necessary
Authenticated user data & metadata
Mote Tx metadata
Add mote (default parameters)
authentication key
Gateway Rx metadata
Gateway status
Mote store

Figure 3: Network server summary dataflow diagram

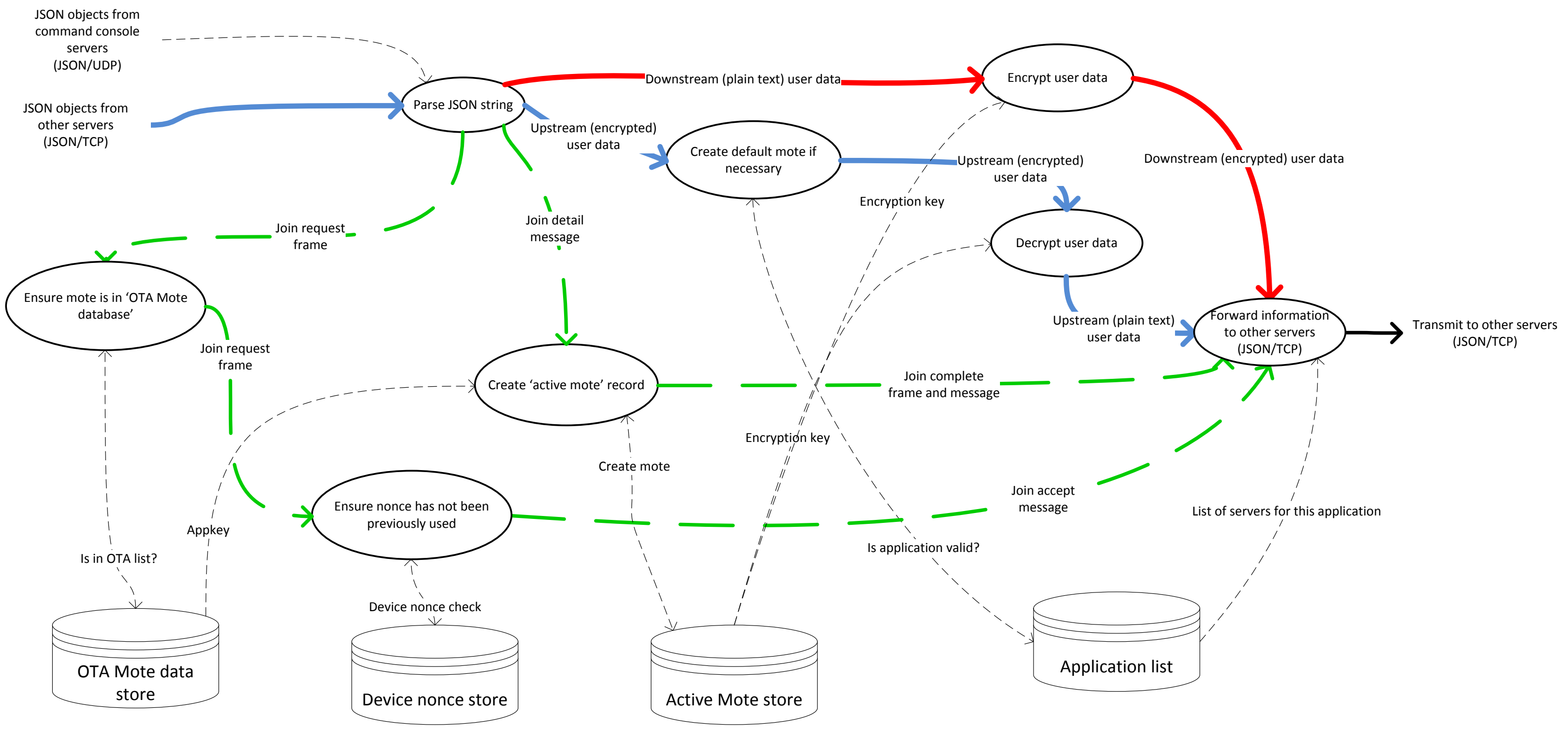

JSON objects from command console servers (JSON/UDP)
JSON objects from other servers (JSON/TCP)
Parse JSON string
Downstream (plain text) user data
Encrypt user data
Upstream (encrypted) user data
Create default mote if necessary
Upstream (encrypted) user data
Downstream (encrypted) user data
Encryption key
Decrypt user data
Join request frame
Join detail message
Ensure mote is in 'OTA Mote database'
Join request frame
Upstream (plain text) user data
Forward information to other servers (JSON/TCP)
Transmit to other servers (JSON/TCP)
Create 'active mote' record
Join complete frame and message
Encryption key
Create mote
Ensure nonce has not been previously used
Join accept message
List of servers for this application
Appkey
Is in OTA list?
Is application valid?
Device nonce check
OTA Mote data store
Device nonce store
Active Mote store
Application list

Figure 4: Application server summary dataflow diagram

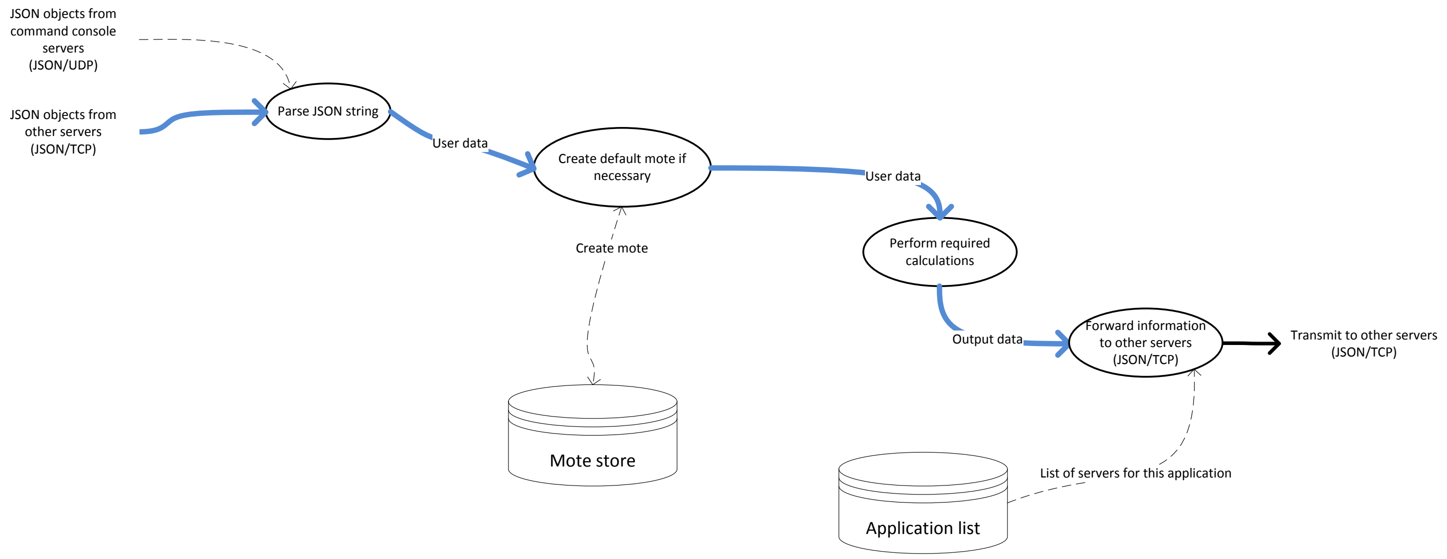

JSON objects from command console servers (JSON/UDP)
JSON objects from other servers (JSON/TCP)
Parse JSON string
User data
Create default mote if necessary
Create mote
Mote store
User data
Perform required calculations
Output data
Forward information to other servers (JSON/TCP)
Transmit to other servers (JSON/TCP)
List of servers for this application
Application list

Figure 5: Network controller summary dataflow diagram

4.2 Description

4.2.1 Network server

'Parse GWMP packet' receives UDP packets from the gateways. It parses the payload of the UDP packet as a LoRa GWMP message, defined in [2]. If the message is successfully parsed, the transform immediately sends a GWMP acknowledgement message, also defined in [2], and stores the source IP address of the sending gateway in 'Gateway Status'. It passes the content of the message, which is a JSON string, to **'Parse JSON string from gateway'**.

'Parse JSON string from gateway' receives top level JSON objects and parses them. The received JSON string is expected to contain one or both of the following JSON objects (each is defined in [2]):

stat:	Gateway status information, is stored by the gateway status store and also passed to **'Forward information to other servers'**.
rxpk:	Received LoRa frame and information concerning it (metadata). The frame, and its metadata, is forwarded to **'Parse mote frame'**.

'Parse mote frame' determines whether the received frame is a data frame or a join frame. If the frame is a join frame, it is passed to **'Receive join request frame'**. Otherwise, it is passed to **'Create default mote if necessary'**

'Create default mote if necessary' creates a mote object within the NS if the mote does not already exist and 'default mote creation' is enabled. If the mote does not already exist and 'default mote creation' is disabled, the received frame is discarded; Otherwise the frame is then passed to **'Authenticate frame'**.

'Authenticate frame' attempts to authenticate the incoming LoRa frame. If this fails the frame is discarded. Otherwise the frame (and its metadata) is stored in **'Rx Frame delay'**. If the frame was not authenticated and the mote was automatically created as a result of receive this frame, the mote too is deleted.

'Rx Frame delay' stores the data and metadata of received frames before transmission to other servers. Because many gateways may receive a single transmission from a mote, the network server stores the first copy of the received frame, the first copy of the metadata concerning the mote's transmission and every copy of the metadata concerning the frame's reception by a gateway. The data is held for the configured time Global::gatewayToNetworkServerMaxDelay_ms from the time of receipt of the first copy of the frame. When the delay has expired, the **'Rx Frame delay'** store passes the frame and its metadata to **'Forward information to other servers'**.

'Forward information to other servers' receives a JSON string, an application EUI and the 'service'[1] type of the data. Each application is served by one or more remote[2] LoRa servers. Each application is configured to forward data (of a particular 'service' type) to one or more remote servers. The transform forwards any received data to all of the servers that are configured to receive the data type for the current application.

'Receive join request frame' receives a LoRa 'join request' frame. It reads the frame's application EUI. If the application EUI identifies a record in **'application list'**, the transform passes the frame and its metadata to **'Forward information to other servers'**.

'Receive join accept from AS' receives a 'join accept' message from the AS. The request contains the EUI of the mote. The transform generates the mote's LoRa network address, and adds it to the **'network address to mote EUI mapping store'**. It then sends a 'Join details' message, containing the mote EUI, application EUI, mote LoRa network address and device nonce, to the AS.

'Receive join complete from AS' receives a 'join complete' message from the AS. The message contains the 'join accept' frame that the NS is to send to the Mote via the Gateway and the authentication key that the NS must use to authenticate transmissions to and from the mote.

'Tx opportunity store' holds a list of opportunities to transmit to motes. To conserve power, a mote operates its receiver (by default) for a short period one second after and two seconds after the end of the mote's upstream transmission. The server must therefore instruct the gateway to transmit at this time. When a potential transmission time is approaching, **'Tx opportunity store'** queries the mote object's **'Mote app data TX Q'** to ascertain whether the mote has data to transmit[3].

[1] Service type is described in Section **Error! Reference source not found.**
[2] 'Remote' in this section means another server, which may be executing on the same machine.
[3] The data sent to a mote comprises user data, acknowledgements of received frames, and MAC commands.

'Mote app data TX Q' holds, for each mote:

- User data waiting for transmission to the mote
- MAC commands waiting for transmission to the mote
- Gateway Rx meta data. This data holds, for each receiving gateway, the 'signal quality' of the most recent transmission received from the mote. This information is used to determine the gateway selected for the transmission to the mote.

If a transmission is to be made, **'Mote app data TX Q'** passes the information to **'Form mote frame for transmission in JSON string'**.

'Form mote frame for transmission in JSON string' generates the LoRa frame and encodes it into a JSON "txpk" object (defined in [2]). The JSON object is then sent to the IP port address of the chosen gateway, using GWMP (also defined in [2]).

4.2.2 Application server

'Parse JSON string' receives JSON objects. It passes:

- Downstream user data (normally received from CS) to **'Encrypt user data'**
- Upstream user data (normally received from NS) to **'Create default mote if necessary'**
- Join requests (normally received from NS) to **'Ensure mote is in OTA mote database'**
- Join details (normally received from NS) to **'Create active mote record'**

'Ensure mote is in OTA Mote Database' receives 'join request' frames. If the contained mote EUI and application EUI are not contained in a single entry in **the 'OTA Mote data store'**, the frame is discarded. Otherwise the frame is passed to **'Ensure nonce has not been previously used'**.

'Ensure nonce has not been previously used' receives a 'join request' frame. If the contained mote EUI and device nonce <u>are</u> contained in a single entry in the **'Device nonce store'** the frame is discard. Otherwise the transform creates a new entry in the **'Device nonce store'** and generates a 'join accept' message and sends it to the **'Forward information to other servers'**.

'Create active mote record' receives 'join detail' messages from **'Parse JSON string'**. It creates an 'active mote' record, adds it to the **'active mote store'** and sets the authentication and encryption session keys. The transform then generates a 'join complete' message and sends it to the **'Forward information to other servers'**.

'Create default mote if necessary' creates a mote object within the NS if the mote does not already exist and 'default mote creation' is enabled. If the mote does not already exist and 'default mote creation' is disabled, the received frame is discarded; otherwise the frame is passed to '**Decrypt user data'**.

'Decrypt user data' decrypts user data, using the mote's encryption session key and passes the result to **'Forward information to other servers'** for transmission to other servers.

'Encrypt user data' encrypts user data, using the mote's encryption session key and passes the result to 'Forward information to other servers' for transmission to other servers.

'Forward information to other servers' receives a JSON string, an application EUI and the 'service'[4] type of the data. Each application served by one or more remote[5] LoRa servers. Each application is configured to forward data (of a particular 'service' type) to one or more remote servers. The transform forwards any received data to all of the servers that are configured to receive the data type for the current application

4.2.3 Network controller

'Parse JSON string' receives JSON objects. It passes the parsed data to 'Create default mote if necessary'.

'Create default mote if necessary' creates a mote object within the NC if the mote does not already exist and 'default mote creation' is enabled. If the mote does not already exist and 'default mote creation' is disabled, the received meta data is discarded. Otherwise the meta data is passed to **'Perform required calculations'**.

'Forward information to other servers' receives a JSON string, an application EUI and the 'service'[6] type of the data. Each application served by one or more LoRa servers. Each LoRa server is configured to receive one or more 'service' types. The transform forwards any received data to all of the servers that are configured to receive the data type for the current application.

'**Perform required calculations'** performs the calculations required to implement the control algorithms. These are specific to the algorithm.

downstream:	The remote server is closer, topologically, to the motes than is this server. This allows downstream data that this server must sent to a mote to be sent to the remote server for forwarding to the mote.
maccmd:	The remove server should receive the content of any header extension fields
gwst:	The remote server should receive gateway status information

[4] Service type is described in Section 4 of [1]
[5] 'Remote' in this section includes servers that are executing on this machine.
[6] Service type is described in Section 4 of [1]

5 Glossary

'/':	The construct 'a/b' is used when Protocol 'a' is transported by Protocol 'b'.
ADR:	Adaptive Data Rate. ADR observes the quality of the signal received by the mote and changes the mote's spreading factor and transmit power in order to optimise the time and energy required for the mote to transmit a frame.
Application:	An application is identified by an 'application EUI'. Each mote is assigned to a single application. The remote server or servers to which information is forwarded (for example the AS to which an NS forwards are received frame) are configured for each application.
AS:	The LoRa application server
ASCII:	American Standard Code for Information Interchange. A widely used standard for representing Latin text, Arabic numerals and punctuation as binary values.
Base64:	A method of encoding binary data into ASCII text. The LoRa system uses Base64 to transport LoRa frames in JSON objects. Base64 is defined by IETF RFC 4648 [5].
Command Console:	The LoRa command console is a program that allows a user to configure LoRa servers.
Cryptographic hash:	The generation of a hash code using a key which is known only to the sender and receiver or receivers. The transmission and recalculation of a cryptographic hash can be used to verify that the message content has not changed.
CS:	The LoRa customer Server
dB:	decibel; a logarithmic ratio of power. Defined by Bell Laboratories
dBm	A logarithmic measure of power, decibel, relative to 1mW
Downstream:	Toward the mote
End-device:	Synonymous with 'mote'
EUI:	Extended Unique Identifier. In this document 'EUI' refers to a value from the 'EUI-64' number space managed by the IEEE.
Firewall:	A firewall is a network security system that controls the incoming and outgoing network traffic based on an applied rule set. A firewall establishes a barrier between a trusted, secure, internal network and another network (e.g., the Internet) that is assumed not to be secure and trusted.
Gateway:	A LoRa gateway is transmits LoRa frames to, and receives LoRa frames from, LoRa motes
GWMP:	Gateway message protocol. The protocol used the transport JSON objects between the network server and the gateways. Defined by [2].

IEEE:	Institution of Electrical and Electronic Engineers (www.ieee.org).
IETF:	Internet Engineering Task Force (www.ietf.org).
IP:	Internet Protocol
IP port address	An IP address or host name and either a UDP or a TCP port number. This document represents a port address in the form <IP address>:<port number> or <host name>:<port number>. E.g. 1.2.3.4:4500 or a.com:4500.
Join:	A colloquial name for 'Over The Air' activation.
Join request frame:	A LoRa frame sent as the initial part of the OTA activation protocol. The frame contains the mote's EUI, its application's EUI and a nonce (a 16 bit random number).
Join accept frame	A LoRa frame sent as the concluding part of the OTA activation protocol. The frame contains the mote's LoRa network address, its network Id and the application nonce (a 24 bit random number generated by a server).
JSON:	JavaScript Object Notation. JSON is a textual based method of representing name, value pairs. The value of an object may itself be a JSON object. Within LoRa, JSON objects contain only ASCII characters. It is defined by [4].
JSON object	A JSON name, value pair
Key:	In cryptography, a key is a piece of information (a parameter) that determines the functional output of a cryptographic algorithm or cipher. Without a key, the algorithm would produce no useful result.
LoRa:	Long Range. Defined by the LoRa Alliance
LoRa Alliance:	The industry body that defines the LoRaWAN protocol. (http://lora-alliance.org/)
LoRa port:	Any user data transmitted to or received from the mote is associated with a 'port' number. User data to or from LoRa Port 0 is MAC command or MAC status data. The remaining 255 LoRa port values are available to the mote user.
LoRaWAN:	The protocol by which a LoRa mote communicates with a LoRa gateway. LoRaWAN is defined by the LoRa Alliance [1].
MAC:	Media access control
MAC command:	A command transmitted to the mote. A MAC command is transmitted to the mote either in the LoRa frame 'header option' area or as user data to LoRa Port 0. Multiple commands may be transmitted in a single frame.
MAC status:	Status information received from the mote. A MAC status message is transmitted by the mote either in the LoRa frame 'header option' area or as user data from LoRa Port 0. Multiple status messages may be transmitted in a single frame.

Metadata:	LoRa Metadata refers to information about the transmission or reception of a LoRa frame.
Mote:	A LoRa end device. A LoRa mote communicates with a LoRa Gateway using the LoRa MAC or LoRa WAN protocol.
NC:	The LoRa network controller
Network id:	The 'network id' of a mote is its 'network address' shifted right by 25 bits, leaving 7 bit value.
Network address:	The LoRa network address is a 32 bit value contained in the LoRa frame that identifies its source or destination mote. The network address need be unique only within the transmission range of a mote or gateway and is distinct from the mote EUI.
NS:	The LoRa network server
OTA:	Over the Air
Over the air:	One of two methods of adding a LoRa mote to a LoRa network. In the OTA method, the mote is configured with a mote EUI, an application EUI and a 128 bit cypher key ('appKey'). Handshaking between the mote and the LoRa servers causes a 32 bit LoRa network address and two 128 bit session keys to be generated. One session key (the 'authentication' key) is known to the mote and the NS. The other (the 'encryption' key) is known to the mote and the AS.
Process:	A running computer program. A process cannot access the memory used by another. Processes are started and stopped independently of others.
Personalization:	One of two methods of adding a LoRa mote to a LoRa network. The mote is configured with its network address and its authentication and encryption keys. The mote's EUI is always equal to its network address and the application EUI is always zero.
Provisioning:	A synonym for 'personalization'
RSSI:	Received Signal Strength Indication. The power of the received signal, normally measured in dBm.
Rx:	Receive
Suspend:	A thread is suspended when it is not available to execute because it is waiting for an event to occur.
Signal quality:	The signal quality is normally measured in dBm and is the sum of the SNR (measured in dB) and the RSSI (measured in dBm).
SNR:	Ratio of signal power to noise power, normally measured in dB.
Spreading factor:	A parameter of a LoRa transmission. Two to the power of 'spreading factor' 'on the air' bits are transmitted to represent each frame bit.

TCP:	Transmission Control Protocol. A connection based protocol for transporting a sequence of bytes. While the connection exists, the content is guaranteed to be delivered in order and without loss or corruption.
Thread:	An independent path of execution within a process. The threads of a process share access to memory within the process.
Transform:	An element of a data flow diagram that transforms its inputs to generate one or more outputs (http://en.wikipedia.org/wiki/Data_flow_diagram)
Tx:	Transmit
UDP:	User Datagram protocol: a simple protocol for transporting data packets. Delivery is not guaranteed. In addition the order of receipt is not necessarily the same as the order of transmission.
Wake:	A thread 'wake' reverses the action of 'suspending' a thread
upstream:	Away from the mote
UTC	Co-ordinated Universal Time; also known as Greenwich Mean Time and Zulu

6 References

Each trademark is the property of its owner.

[1] Semtech Ltd, “LoRaWAN Network Server Demonstration: High Level Description,” 2015.

[2] Semtech Ltd, “LoRaWAN Network Server Demonstration: Gateway to Server Interface Definition,” 2015.

[3] IETF, “The Base16, Base32, and Base64 Data Encodings,” October 2006. [Online]. Available: https://www.ietf.org/rfc/rfc4648.txt.

[4] ECMA International, The JSON Data Interchange Format, 2013.

[5] LoRa Alliance, “LoRaWAN Specification,” LoRa Alliance, 2015.

Contact Information

Semtech Corporation
Wireless Sensing and Timing Products Division
200 Flynn Road, Camarillo, CA 93012
Phone: (805) 498-2111 Fax: (805) 498-3804
E-mail: support_rf_na@semtech.com
Internet: http://www.semtech.com