LoRaWAN® Roaming Hub Technical Recommendations (TR10-1.0.0)

NOTICE OF USE AND DISCLOSURE

LoRa Alliance®
5177 Brandin Court
Fremont, CA 94538
United States

LoRaWAN® Roaming Hub Technical Recommendation (TR10-1.0.0)

Authored by the LoRa Alliance Technical Committee

Technical Committee Chair and Vice-Chair:
A. YEGIN (Actility), O. SELLER (Semtech)

Editor:
A. YEGIN (Actility)

Contributors:
S. BALAKRICHENAN (AFNIC), R. LORRAIN (Semtech), B. MARKUS (LORIOT), A. YEGIN (Actility)

Version: 1.0.0
Date: Jan 25, 2022
Status: Final

Contents

Figures

1 Conventions

The keywords "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in IETF Best Current Practice 14 (BCP14 [RFC2119] [RFC8174]) when, and only when, they appear in all capitals, as shown here.

The tables in this document are normative. The figures in this document are informative.

Document titles are written as *LoRaWAN Link Layer Specification.*

Commands/messages are written ***PRStartReq*** and fields are written `DevAddr`.

2 Introduction

The LoRaWAN roaming feature, as defined in the *LoRaWAN Backend Interfaces Specification* [LWBE], requires Network Servers (NSs) and Join Servers (JSs) to be configured to process incoming messages from their peers and route those messages to their respective destinations, based on roaming agreements. Configuration parameters include Uniform Resource Identifiers (URIs) or IP addresses corresponding to JoinEUIs (IDs identifying a Join server) and NetIDs, firewall rules (e.g., allowing incoming TCP connections from given IP addresses), HTTP authentication credentials, LoRaWAN roaming policies (e.g., allowing passive roaming, incoming roaming devices, etc.), and definition of Usage Data Records (UDRs).

As the number of NS and JS nodes involved in roaming increases, managing the exponentially increasing number of peer-to-peer configurations, along with the associated record keeping/reporting, has become difficult to set up and maintain.

This is traditionally where a Roaming Hub (RH) comes into play. An RH helps transform the interconnection among several NSs/JSs from a mesh topology to a star topology by inserting itself between any two nodes, as depicted in Figure 1. Such a change in topology helps reduce the complexity of roaming management from $O(n^2)$ to $O(n)$.

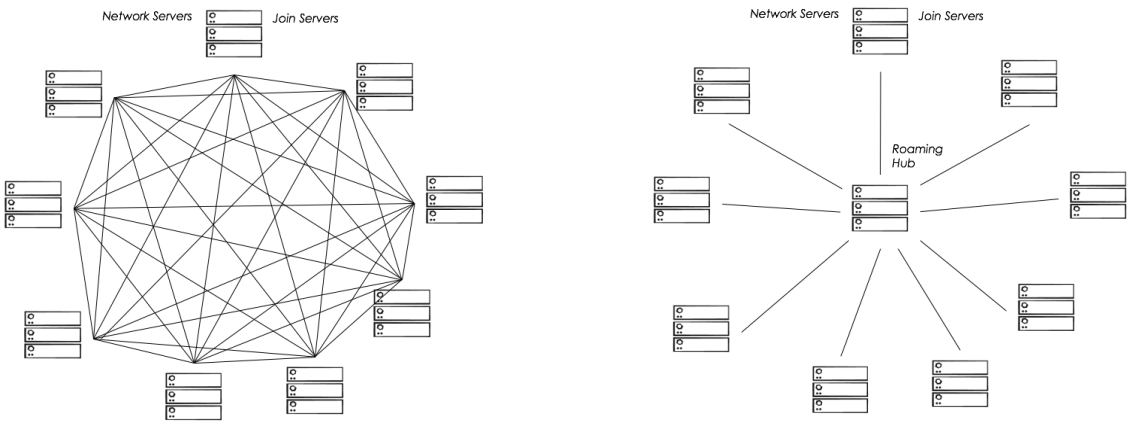


Figure 1: Transforming a mesh topology into a star topology with the help of an RH

This document describes various aspects related to the implementation and operation of RHs, in the form of a LoRa Alliance technical recommendation. There are no new RH-related backend message types or message fields introduced in this document. All messages discussed in this document are defined in the the *LoRaWAN Backend Interfaces Specification* [LWBE].

3 Functionality

There are two primary functionalities of a roaming hub:

1. Forward messages among peers based on the roaming agreements – The RH forwards an incoming message to its ultimate destination only if there is an associated agreement that allows it to do so. Otherwise, the RH rejects the incoming message, just like the intended destination would do.
2. Keep track of and report the usage, in the form of UDRs – The amount of traffic passed among the nodes served by the RH is recorded in a peer-to-peer fashion and is reported to each node at the end of a configured period.

One or more roaming hubs gets inserted between NSs and JSs, acting as a transparent stateless proxy for the nodes it is serving, as depicted in Figure 2. This example illustrates a single roaming hub.

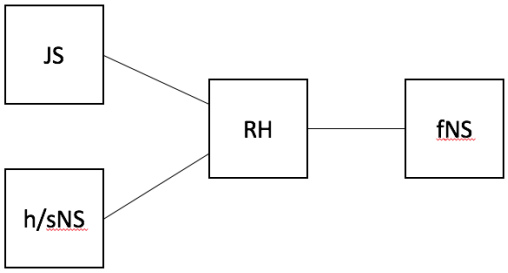


Figure 2: Single roaming hub insertion

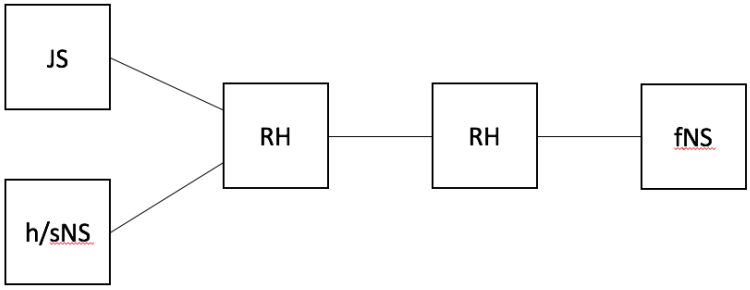
When a node needs to deliver a message to its peer, it sends the message to the RH instead of directly sending to the peer. For example, the Forwarding Network Server (fNS) sends a ***HomeNSReq*** message (to query the home network server of a particular device) that is ultimately destined to the JS towards the RH, so the RH can deliver that message to the JS. Similarly, the Home Network Server (hNS) sends a ***XmitDataReq*** message that is ultimately destined to the fNS towards the RH, so the RH can deliver that message to the fNS.

The RH appears as a JS when receiving messages from the fNS for the JS. And, the JS treats the RH as an fNS when sending messages to the RH, to be ultimately delivered to the fNS. Therefore, the interface between the JS and the RH, and the interface between the RH and the fNS, is the JS-vNS interface defined in the the [LWBE].

Similarly, the interface between the Serving Network Server (sNS) and the RH, and between the RH and the fNS, is the sNS-fNS interface defined in the [LWBE].

Consequently, the [LWBE] specification is sufficient to implement an RH that is interoperable with the NSs and JSs that are compliant with that specification.

Figure 3 depicts two roaming hubs between the nodes:



165
166 **Figure 3: Multiple roaming hub insertion**

167

4 Message Flows

Figure 4 depicts the call flow that SHOULD be used for the RH forwarding ***PRStartReq*** and ***PRStartAns*** messages between the fNS and the sNS.

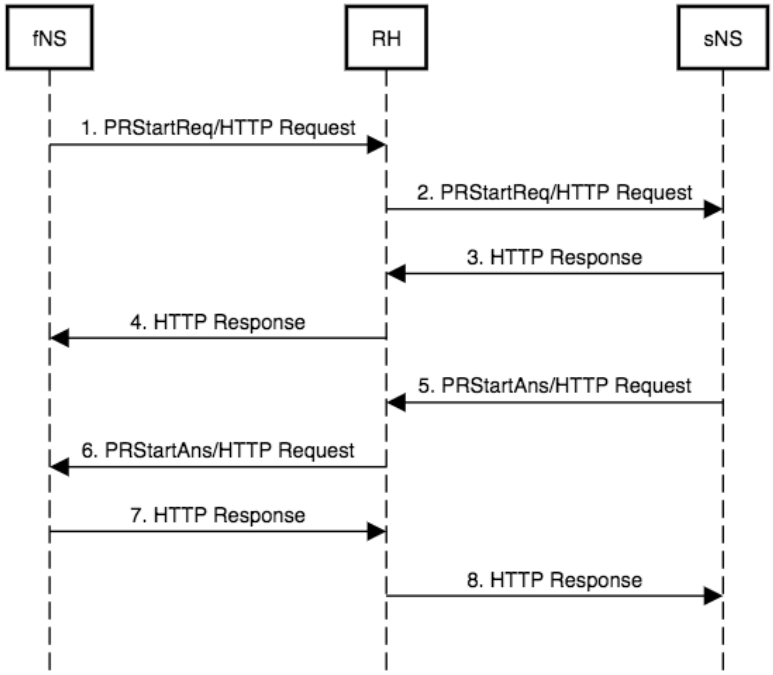


Figure 4: Message forwarding via the RH

The steps in this call flow are described below:

Step 1:
Upon receiving an uplink frame, the fNS generates a passive roaming start request ***PRStartReq*** message. The fNS inspects the end-device address (`DevAddr`), and identifies the NetID of the intended sNS. The fNS identifies the RH as the next hop for delivering backend messages to the sNS. Therefore, the fNS sends the ***PRStartReq*** message to the RH.

Step 2:
The RH inspects the `SenderID` and `ReceiverID` of the incoming ***PRStartReq*** message. The RH forwards the message to the sNS, assuming the RH is configured with a roaming agreement for the NetID pair of fNS-sNS. The sNS processes the incoming ***PRStartReq*** message as if it direcly came from the fNS.

Step 3:
The HTTP server on the sNS sends the HTTP response message to the RH in response to the incoming HTTP request from Step 2.

Step 4:
The RH accepts the incoming HTTP response for its pending HTTP request and generates the HTTP response for the incoming HTTP request at Step 1.

Steps 1-4 complete successful delivery of the ***PRStartReq*** message sent from the fNS to the sNS via the RH.

The ***PRStartAns*** message follows the same path in reverse in a similar way.

Step 5:
The sNS generates a ***PRStartAns*** message. The sNS identifies the RH as the next hop for delivering backend messages to the fNS. Therefore, the sNS sends the ***PRStartAns*** message to the RH.

Step 6:
The RH inspects the `SenderID` and `ReceiverID` of the incoming ***PRStartAns*** message. The RH forwards the message to the fNS, assuming the RH is configured with a roaming agreement for the fNS-sNS pair. The fNS processes the incoming ***PRStartAns*** message as if it direcly came from the sNS.

Step 7:
The HTTP server on the fNS sends the HTTP response message to the RH in response to the incoming HTTP request from Step 6.

Step 8:
The RH accepts the incoming HTTP response for its pending HTTP request, and generates the HTTP response for the incoming HTTP request from Step 5.

Figure 5 depicts the call flow that SHOULD be used when the RH rejects the ***PRStartReq*** message sent by the fNS due to lack of a roaming agreement between the fNS and the intended sNS.

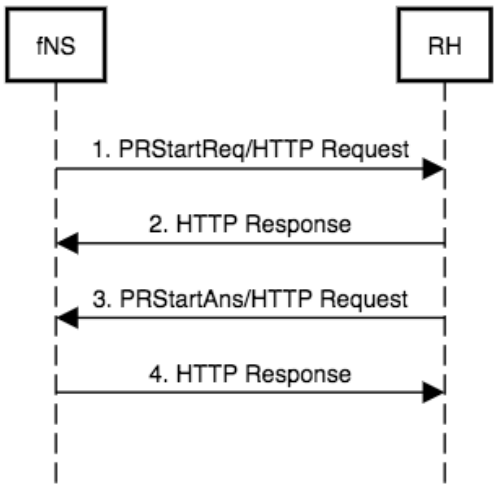


Figure 5: Message rejected by the RH

The steps for rejection are described below:

Step 1:
The fNS sends the *PRStartReq* message to the RH.

Step 2:
The RH inspects the `SenderID` and `ReceiverID` of the incoming ***PRStartReq*** message. The RH decides to reject the message due to lack of a roaming agreement between the fNS-sNS pair and the HTTP server on the RH sends the HTTP response message to the fNS in response to the incoming HTTP request from Step 1.

Step 3:
The sNS sends the ***PRStartAns*** message with `Result=NoRoamingAgreement` to the fNS.

Step 4:
The HTTP server on the fNS sends the HTTP response message to the RH in response to the incoming HTTP request from Step 3.

5 Configuration

Configuration for network servers using an RH SHOULD include the following:

- The firewall must be set to allow HTTP/TCP/IP communication between the NS and the RH.
- There must be an HTTP credential setting to enable secure HTTP communication between the NS and the RH.
- For each roaming partner of the NS that is reachable via the RH, the URL of the RH must be used as the next hop for reaching that roaming partner (as opposed to the roaming partner's actual URL).

Configuration for each RH SHOULD include the following:

- The firewall must be set to allow HTTP/TCP/IP communication between any served NS and the RH.
- There must be an HTTP credential setting to enable secure HTTP communication between any served NS and the RH.
- For each served NS, the RH needs to know:
    - Pair-wise roaming information. This includes NetIDs of the pairs and whether or not passive roaming activation is enabled.

> **Note**: The roaming relationship is directional. For example, NS1's end-devices may be allowed to roam into NS2's network, but NS2's end-devices may not be allowed to roam into NS1's network.

    - Forwarding information, for when the RH needs to send a message to the NS. This includes the URLs used to forward messages destined to the NetID of the NS and whether or not the NS is directly connected to the RH or is reachable via another RH.

6 Usage Data Records

The RH SHOULD keep track of the hub usage in terms of number of uplink and downlink frames forwarded from one NS to another and SHOULD report that information to each associated party at regular intervals (e.g., monthly).

Glossary

fNS	Forwarding Network Server
hNS	Home Network Server
HTTP	HyperText Transfer Protocol
hNS	Home Network Server
IP	Internet Protocol
JS	Join Server
LoRaWAN®	Long Range network protocol
NS	Network Server
RH	Roaming Hub
sNS	Serving Network Server
UDR	Usage Data Record
URI	Uniform Resource Identifier

Bibliography

References

[LWBE] TS2-1.1.0 LoRaWAN Backend Interfaces Specification (version 1.1), LoRa Alliance, November 2020.