LoRaWAN – simple rate adaptation recommended algorithm

Common to Class A/B/C specification

Table of Contents

Revisions

22/1/2014 : ns	Initial release
25/2/2014 : ns	Added the channel mask handling in ADR algorithm
13/10/2016 : ns	Completely revamped algorithm , stop taking into account lost frame to calculate DR. instead use SN and set repetition rate based on packet loss rate to avoid run to the bottom. Also described ADR command transmission periodicity

Abbreviations & Acronyms

ADR	Adaptive Data Rate
LoRa™	Long Range
RF	Radio Frequency
RSSI	Received Signal Strength Indicator
Rx	Receiver
SF	Spreading Factor
SNR	Signal Noise Ratio

Introduction

Rate adaptation is an essential feature of the LoRaWAN. All end-points have the ability to dynamically change the data rate they use for their uplink & downlink transmissions. When the device Adaptive Data Rate mode is enabled, the data rate used is dictated by the central network server through a dedicated ADR (Adapt data rate) MAC command documented in the LoRaWAN specification document.

The server decides to send an ADR command to a given node based on the analysis of history of the received signal quality of this node.

The goal of the present document is to present a simple baseline recommended way to implement this decision mechanism. This does not claim to be the reference or optimal implementation. The actual implementation of a good quality Adaptive Data Rate algorithm is an essential part of any commercial network management service.

Known limitations

- This algorithm is only suitable for STATIC end-devices. Adaptive Data Rate should not be used on mobile end-devices because the radio channel changes dramatically with every frame
- This algorithm should not be used if a device needs to be located though the network using LoRaLOC TDOA. For localization to work the device should be received by at least 3 (preferable 4) gateways. This algorithm scales up data rate aggressively and does not take into account gateway receive diversity constraint as an input.
- This ADR algorithm does not make use of SF7/250kHz and FSK50kbit/sec rates, it is limited to SF12/125kHz to SF7/125kHz (DR0 to DR5 in LoRAWAN).
- Does not measure packet loss rate per channel but globally. Therefore this algorithm is not suitable to allocate different channels based on a targeted Service Level Agreement.
- This algorithm in its present form is limited to EU868 ISM band operation because it specifically refers to data rates defined in the EU868 section of the LoRaWAN regional channel definition. Its extension to other regions is work in progress.

Algorithm inputs

Each time a node performs an uplink transmission, the frame may be received by several gateways. Inside each of the gateways, the frame is tagged with its signal strength (RSSI) and Signal to Noise ratio in dB (SNR). The gateways then forward the tagged frame to the network server.
So for a given frame the server will receive several copies, each coming from a different gateway and tagged with different RSSI and SNR values.
For a given frame, the notation {gatewayID , frame_nb, SNR} represents the SNR measured by gateway ID.
For example if a frame is received by 2 gateways A and B , with corresponding SNRs , SNRa and SNRb , the following information is available in the server :[(A,SNRa) , (B,SNRb)] . The values SNRa and SNRb are values directly reported by the gateway HAL layer as provided by Semtech without any further processing.

Each frame features a 16 bits frame counter (F) field which is incremented with every transmission.
So for each transmission of a given node (DevAddr) we define the notation SNRmax(devAddr,F) where:

- F is the value of the frame counter of the frame
- DevAddr is the network address of the node as defined in the LoRaWAN spec
- SNRmax is the maximum of the various SNRs reported by the different gateways who received this given frame.

Continuation of the previous exemple:
The frame coming from node DevAddr with frame counter F was received by gateways A&B with respective SNRs SNRa & SNRb where SNRa > SNRb. Then SNRmax(DevAddr,F) = SNRa

For each node taken into account by the algorithm, the following inputs must be available in memory.

1. The frame counter values of the last 20 received frames F_n, F_{n-1} .. , F_{n-19}
2. The list of the 20 associated SNRmax(DevAddr, $F_{n \text{ to } n-19}$)
3. The number of gateway which received each of the frame (the receive diversity) GtwDiversity(devAddr,$F_{n \text{ to } n-19}$) [not yet used by this algorithm , reserved for future use]

Example: The following data structure is available for the last uplinks of node 123

Node : 123

Frame counter	SNRmax (dB)	GtwDiversity
10	-6	2
11	-7	2
12	-25	1
13	-25	1

14	-10	2
16	-25	1
17	-10	2
19	-10	3
Etc…		
29	-7	2

Note: some frames may be lost, in our example frames 15 & 18

Each time a new frame with frame counter N+x reaches the server, the SNRmax(Nif,F_{n-x}) value is computed by analyzing the multiple instances of the frame reported by the different gateways. The triplet {N+x , SNRmax(devAddr,F_{n-x}) , GtwDiversity(devAddr,F_{n+x})} is inserted at the end table. Each time a new line is inserted at the end of the table, the lines 2 to 19 are shifted up and the first line is discarded so as to keep the length of the table constantly equal to 20. If a frame is retransmitted several time by the end-device (same frame counter) only the best transmission is kept. That means that if a frame with the same frame counter value than the newly received frame already exists in the table (it can only be the last one) then the SNRmax value is updated if the new frame SNRmax value is greater than the one already stored in the table.

Optimal Rate determination Algorithm

Each time the server decides to send an ADR command to a node, it uses the previously described data structure to perform the following computations.
First the **Max** (not average) SNR (SNRm) is computed over the middle column of the table. Then we compute:

SNRmargin = SNRm – SNR(DR) - margin_db

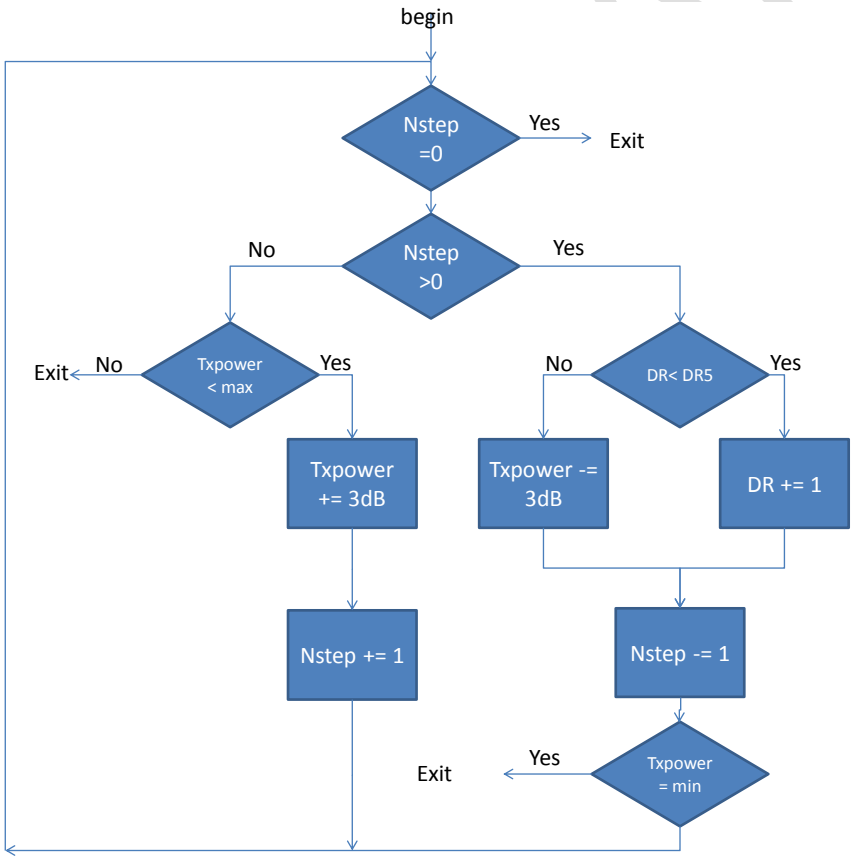
- Where margin_db is the installation margin of the network (typically 10dB in most networks) , this is a device specific static parameter. This may be part of a device profile.
- Where SNR(DR) is the required SNR to successfully demodulate as a function of Data Rate given in the following table. DR is the data rate of the end-device's last received frame.

Data Rate	Required SNR
DR0	-20
DR1	-17.5
DR2	-15
DR3	-12.5
DR4	-10
DR5	-7.5

Note: The default recommended margin_db value is 5dB. Increasing margin_db on all devices tends to globally lower the data rate used ,and therefore reduces the overall network capacity but increases the link margin and will tend to lower the packet loss rate in lightly loaded networks. On the other hand, lowering margin_db , tends to push data rates up , increases network capacity, but will increase packet loss rate. Again, margin_db can be set per device depending on the device profile.

Note: The SNR estimator considered is the Max over the last 20 frames received because in this algorithm we consider that the main packet loss mechanism is interference with other ISM band devices and also with other devices of the same network. In that situation it is better to use max() as estimator and subtract a margin rather than using the average SNR as estimator.

Then we compute NStep = int(SNRmargin/3) , where *int* is the integer part which can be negative or positive. *(example int(-3.8) = -3 , int(3.8) = 3)* The next flow diagram illustrates the step of the algorithm to follow.



Note: In the case where Nstep is negative and TXPower is already at its maximum the algorithm does not try to lower the data rate, because the end-device's implements automatic data rate decay. The algorithm can only actively increase the data rate. Trying to lower the data rate leads to constantly oscillating values.

Optionally, the device profile might also include a DRMin field. In that case if the device uses a Data Rate lower than DRmin , the server shall send an ADR command setting Data Rate to DRmin.

Optimal transmit repeat rate determination

Through the same ADR MAC command the network server also instructs the end-device to repeat each transmission from 1 to 15 times (NbRep field of the linkADRReq MACcommand)
The number of repetition is computed from the packet loss rate.
PktLossRate(devAddr) is computed for the left column (FCounter) of the table.
PktLossRate(devAddr) = (FCounter(last) – Fcounter(first) -20) / (FCounter(last) – Fcounter(first)) .
where *Fcounter(last) is the value of FCounter contained in the last line of the table, (resp FCCounter(first) in the first line)*

The following table gives the the new NbRep parameter as a function of the currently NbRep value and the measured PktLossRate to target roughly 10% packet loss rate. This table may be changed to target lower or higher average packet loss rate.

Current NbRep / PktLossRate	1	2	3
<5%	1	1	2
5 to 10%	1	2	3
10 to 30%	2	3	3
>30%	3	3	3

Note: this algorithm limits the maximum number of Tx repetitions to 3. This is sufficient in the vast majority of use-case studied until now.

ADR update periodicity

How often the network actually sends ADR MAC command to the end-device's depends mainly on the goal to achieve, optimal network capacity , lowest packet rate or simplicity.

The simplest strategy is to systematically send an ADR command to the end-device each time the device as set the ADR_ACK_Req bit in the FCtrl frame field of its uplink. The device will signal that it has successfully received the LinkADRReq command by responding with a LinkADRAns command , by unsetting the ADR_ACK_Req bit and eventually by changing its data rate and/or tx power . Those 2 mechanisms (linkADRAns and ADR_ACK_Req bit) are provided because:

- LinkADRAns provides usefull information on why an end-device might reject an ADR command, but the frame containing the LinkADRAns might be lost
- ADR_ACK_Req bit is present in every uplink frame , therefore this bit switching back to 0 signals that the ADR command was received no matter how many uplink frames were lost.

A slightly more elaborated strategy is to send an ADR command to the end-device at most every Nth uplinks and only if the device's data rate can be increased or if its TX power can be lowered. In the absence of ADR command control the device will autonomously first increase it TX power back to nominal then decrease the data rate.

Channel mask

The ADR MAC command can also be used by the server to limit an end-device to a given set of frequency channels amongst all the usable channels.

This strategy might be used by a network server to reserve some channels for certain classes of applications (or certain device profiles). However if a channel suddenly becomes unusable (due to a temporary interferer in one of the network cell for example), the end-devices limited to this precise channel might not be able to regain connectivity even at the lowest data-rate. It is therefore not recommended to constrain a device to a single channel. The device should preferably be left with the possibility to frequency hop across at least 2 channels.

Therefore when a node performing Adaptive Data Rate reaches its minimum data rate, it will (as specified in LoRaWAN1.0.2 & further) re-enable all its defined channels.