ClAir Sampling and Transmission Scheme

Ideally, sensor nodes on the ClAir network measure data frequently and transmit these samples as soon as they become available. In reality, though, we have to adhere to a number of constraints. In the following, we derive when to measure and how to transmit the measurements to the ClAir Server.

Design Goals

Goals of the ClAir network are to

- inform the general public about current and past air quality in publicly-accessible indoor locations,
- and to provide timely air quality information to location operators.

Technically speaking, we want to achieve both *low latency* and *high resolution*. Latency is the duration between the instant a node takes a measurement and the instant the ClAir Server makes this measurement available via its API. Resolution is inversely proportional to the *sampling interval*, the time between two measurements.

The lower the latency, the more current the information for on-the-spot decisions and reactions: Should I enter this store? Should I open the windows now? Resolution, on the other hand, is important to see changes in air quality over time, and thus to trust the data in the long run: Are spikes of high CO2-concentration visible in the resulting graph? Does the week-view show that I regularly ventilate the room?

Simulations of the air circulation in classrooms show that air exhaled by a supposedly infected person will have dispersed throughout the room in about 20 minutes. Even though a classroom is not representative for other indoor locations, we can determine that both latency and sampling interval should be shorter. On the other hand, air quality does not change on the order of seconds. Therefore, we do not need to push latency and sampling interval below one minute.

The ideal sensor node, therefore, should measure the air quality every minute and immediately transmit the resulting measurements to the ClAir Server. However, there are technical restrictions; in particular, the LoRaWAN wireless transmission technology and The Things Network (TTN) as our primary LoRaWAN provider have limitations that require compromises, as discussed next.

Technical Restrictions

LoRaWAN is a low-power wireless technology developed especially for sensor networks.

Packet Airtime

LoRaWAN uses an adaptive modulation and coding scheme (MCS) that trades off data rate for transmission range: The slowest MCS adds so much redundancy that the receiver can reliably demodulate and decode the received packet even if the reception level is very weak. Consequently, the packet occupies a radio channel for a long time: A message with a mere 3 bytes payload may take 25.7ms via the fast MCS SF7 on a 250kHz radio channel, but 1319.9ms using the slow but robust SF12 on a 125kHz radio channel.

When we refer to a *message* here, we mean the *application payload* part of a LoRa message. LoRa adds another 11 bytes of protocol header. Because of the way message and header are encoded, the actual airtime is nonlinear in the message size. For each MCS, the channel encoder operates on fixed blocks of input bits. If the input message is shorter, it will be padded.

TTN Fair Use Restrictions

The unlicensed 868MHz band is a shared resource. Therefore, EU regulation restricts the duty cycle of any transmitter to 1%. Furthermore, the not-for-profit TTN has a Fair Use Policy (FUP) to equally distribute the scarce gateway and network resources:

- The total *airtime* per Node is limited to 30s within 24h on the uplink (Node to gateway).
- Because a gateway cannot receive while transmitting, only 10 downlink messages are admissible per node per day.

Design Considerations

The TTN FUP restrictions have a profound impact on how often ClAir Nodes may take measurements and transmit data. The ideal case of one measurement transmitted per minute would lead to 1.440 uplink messages per day. A node that uses the slow SF12 would require a total airtime of more than 31 minutes per day, instead of the admissible 30 seconds. And one in 140 messages only could be acknowledged via a corresponding downlink message. Several design trade-offs are necessary to arrive at a transmission scheme that is admissible and viable. In the following, we detail our design considerations and design decisions.

Encoding Efficiency

Compared with WiFi or 4G networks, LoRaWAN offers very low data rates. Messages typically have a few bytes of payload only. The largest message possible on TTN has 51 bytes payload at SF12 and 222 bytes at SF7. Widely used data compression schemes, such as hte Lempel-Ziv-algorithm, are not effective for messages this short. Therefore, it is very important to parsimoniously encode the measurement values with the fewest bits possible.

The Sensirion SCD-30 sensor on our prototype nodes measures CO2, temperature, and relative humidity. We manage to squeeze the three measurement values into two bytes:

- One byte for the CO2 concentration, quantized at 20 parts per million (PPM) from 0PPM to 5,100PPM.
- Five bits for the temperature from 0°C to 31°C.
- And three bits for relative humidity quantized to 10% from 10% to 90%.

Other node models with other sensors might require a different encoding.

Dynamic Adaptation of the Modulation and Coding Scheme

The easiest way to satisfy the airtime constraint would be to use the fastest MCS only, so that messages are short. But this would reduce transmission range and thus the utility of the ClAir network. We want to cover the largest distance possible, to benefit the largest number of businesses and organizations. Because we cannot know up front where nodes will be placed, we need to activate the adaptive data rate (ADR) feature on the nodes, where the TTN Network Server dynamically selects the fastest MCS that still guarantees stable reception. Consequently, ClAir Nodes need to support all MCS, from SF7/250 to SF12/125.

Latency and Resolution

Because of the fixed protocol overhead of 11 bytes per message, it is more airtime-efficient to collect multiple measurements and transmit them as one message, instead of transmitting each measurement separately. Such multi-sample messages increase latency, especially for the measurements taken early; on the other hand, multi-sample messages allow us to increase resolution.

The transmission interval, which determines latency, is always an upper bound on the sampling interval. Latency and sampling interval coincide if we transmit one sample per message. Therefore, we first minimize the latency of a one-sample message for each MCS, given the TTN airtime constraint. Then, we increase the number of samples per message as long as the airtime stays constant. Finally, we check the sensitivity: How much would latency increase for additional payload bytes and thus a reduced sampling interval?

Illustration for 2-Byte Samples

In the following, we illustrate this optimization scheme for the 2-byte samples of the ClAirchen prototype node, the encoding of which we already mentioned above.

Analysis

MCS	SF7w	SF7	SF8	SF9	SF10	SF11	SF12
airtime [ms]	26	52	93	165	330	660	1319
payload [byte]	6	6	5	4	6	5	7
min. transmission interval [s]	75	149	267	475	950	1900	3799
... [min]	1,25	2,5	4,5	7,9	15,8	31,7	63,3
#samples	2	2	2	1	2	2	2
sampling interval [s]	38	75	134	475	475	950	1267
... [min]	0,6	1,25	2,2	7,9	7,9	15,8	21,1

To obtain the *airtime* for each MCS, we consult the TTN airtime calculator and increase the *payload* size as long as the airtime stays constant The channel coding that causes this stepwise increase was explained previously.

The *minimum transmission interval* follows from the airtime restriction by division. For a given airtime per message, we derive how many such messages in total are admissible per day. Up to SF10, the transmission interval - and thus the latency - is below the 20 minutes outlined previously. However, the airtime restriction leads to transmission intervals of over 30 minutes and 60 minutes for SF11 and SF12 respectively. Yet, even though the latency is much higher than desired, we do not rule out these MCS: Better to receive data at all than to exclude the corresponding locations up front.

For each message, we assume a header of one byte, and a consecutive range of two-byte samples. With this figure, we can calculate the number of *samples* that can be packed into the message of a given duration. Most messages can accomodate two samples, except for SF9.

Finally, the effective sampling interval results by dividing the transmission interval by the number of samples per message. At SF12, we miss the target of a 20-minute sampling interval.

Optimization

The above design is our baseline for further optimization. We can now analyse the sensitivity of transmission interval and sampling interval with respect to larger payloads: Using the airtime calculator, we increase the number of payload bytes until we hit the next nonlinear increase in airtime. Then, we contrast the increased latency with the reduction in the sampling interval.

MCS	SF7w	SF7	SF8	SF9+	SF10+	SF11+	SF12+
airtime [ms]	26	52	93	186	371	742	1383
payload [byte]	6	6	5	7	11	9	11
min. transmission interval [s]	75	149	267	534	1068	2136	4271
... [min]	1,25	2,5	4,5	8,9	17,8	35,6	71,2
#samples	2	2	2	3	5	4	5

MCS	SF7w	SF7	SF8	SF9+	SF10+	SF11+	SF12+
sampling interval [s]	38	75	134	178	214	534	855
... [min]	0,6	1,25	2,2	3,0	3,6	8,9	14,3

For the fast MCS SF7-SF8, an increased payload does not pay off because a sampling intervall of less than one minute is not neccessary. However, for SF9 and up, the added samples increase the latency only slightly but provide a significant reduction in sampling rate.

Implementation

For a ClAir Node to implement the above transmission scheme, it must maintain a timer for the sample interval. Once the node has accumulated number of samples commensurate with the current MCS, it generates an upling message and transmits it.

Ideally, the samples do not contain one-shot measurements taken at the sampling instant but averages over the entire sampling interval. This averaging acts as a low-pass filter that prevents aliasing with the low sampling rate.