LoRaWAN Performance in Generic IoT Scenarios

Article · June 2017

CITATION
1

READS
304

1 author:

Andrew Wixted
Griffith College
33 PUBLICATIONS **705** CITATIONS

SEE PROFILE

LoRaWAN Performance in Generic IoT Scenarios

Andrew Wixted
andrew.wixted@bigpond.com
(reformatted from original 'white paper' posted at www.stream-technologies.com, June 2017)

Abstract

LoRaWAN is an Internet-of-Things (IoT) technology that combines long range low power wireless with a protocol layer that provides networking functionality. The technology has been designed to provide low speed data communications in a number of typical use-cases. This paper investigates the LoRaWAN functionality in these use-cases.

I. Introduction

There are four generic IoT scenarios that cover most LoRaWAN use-cases. Not unexpectedly, since LoRaWAN was designed for this, these scenarios will align with different LoRaWAN classes and modes of operation.

A. Four Generic Scenarios

i. Set and Forget: A sensor that is designed to transmit periodic sensor data and does not expect any incoming messages from the network.
ii. Smart Sensor: A sensor that can be reprogrammed over the network and therefore does expect messages from the network but relatively rarely and only in response to a sent message (LoRaWAN Class A devices) This could include some actuator devices but most likely this is a battery powered sensor device.
iii. Actuator / Sensor: A device that is designed to control and monitor some physical system which operates in real time. This system expects messages from the network at anytime and is likely to be powered. (LoRaWAN Class C devices)
iv. Groups of Actuator / Sensors that are controlled together. For example Street Lights. In this scenario a single command send from the network can turn on (or off) all the street lights in a specific group.

This paper measures and reports on aspects of all of the above scenarios with three different experiments:

i. Class A devices engaged in two-way communications with an MQTT connected host system (sensor initiated communications)
ii. Class C device engaged in two-way communications with an MQTT connected host system (host initiated communications)
iii. Pseudo multicast, a group of Class C devices simultaneously receiving communications from a host system.

While Class A send-and-acknowledge (ACK) performance has been reported previously, for this experiment the Network is responding exclusively in the Rx-1 window where the previous reported experiments used Rx-2. This means that the downlink performance (Network to Node) for different spreading factors and channels has been measured.

B. Network Topology

The setup was similar across all experiments with minor software variations for the different scenarios. The basic networking is shown in Fig.1. LoRaWAN sensors were Multitech mDot 868MHz devices operating standard AT command embedded software. These were controlled by a Python script running on a Raspberry Pi. Gateways and Network Servers were Stream Technologies Glasgow LoRaWAN Network. The Host System was MQTT connected browser software using MQTT over websockets. The Python scripts or Javascript host program varied for each experiment.

The geographic layout for the segment of the network used in this test was essentially an isosceles triangle with the longs sides approximately 2.15km and the base 1.6km. The base had three gateways with the centre being the top of a hill and the other two down either side. Another gateway was at the apex of the triangle and the test system was within 100 metres of this gateway. One of the base gateways was over a hill in a different reception zone but did appear in the results of experiment 1 (LoraWAN Class-A).

From here on, each experiment and its results are contained within their own section. The layout of the sections is as follows:

Class A Two Way Communications
  Experiment Design
  Experiment Results
  Experiment Discussion
Class C Two Way Communications
  Experiment Design
  Experiment Results
  Experiment Discussion
Pseudo Multicast
  Experiment Design
  Experiment Results

Experiment Discussion
Conclusions
Appendix: De-duplication Error
Glossary of terms and abbreviations

II. Class A devices engaged in two-way communications with an MQTT connected host system (Rx-1 window)

At this time it is expected that the majority of devices that come under the umbrella of the Internet of Things (IoT), and that are expected to be connected via Low Power Wireless, will be battery powered sensors where the value of the sensor and the value of the data from an individual sensor will be low. The benefit comes from being able to capture data from a large number of end points at low cost and then create value and knowledge from the consolidated data. For instance, a city that knows the status of ten thousand public parking spaces and that can supply that information in real time to drivers could reduce congestion and fuel use as drivers can navigate directly to an available space. The sensor only needs to detect a change of status and pass that information to the parking system.

Other smarter sensors may need to interact with their management system. A sensor with a real time clock might require periodic time synchronisation messages. Multi-sensor systems might have a management system that wants to turn sensors on and off or change the sampling interval or reporting interval depending on measurements received. These systems need two-way communications but may not have the power availability to use LoRaWAN Class-C communication.

In LoRaWAN Class-A operation a LoRaWAN sensor can transmit an uplink packet to the network and not request an ACK or the sensor could request an ACK. If an ACK was requested the sensor will listen for that ACK at specific times and on specific frequencies and at a particular spreading factor (SF). The default values of SF and frequency are region specific, the following values are for European systems. The two specific listening times are known as Rx-1 and Rx-2. By default Rx-1 occurs at exactly one second after the uplink message was received by the gateway. Rx-1 ACKs will be made on the same channel as the uplink message and with the same SF. By default the Rx-2 window occurs at exactly two seconds after the original message was received and will use SF12. The frequency is dependent on the jurisdiction but on European systems the Rx-2 ACK will be on 869.525MHz

The default timing can be overridden where necessary, provided both the sensor and the network server are capable of handling alternate timing. Any variation in timing would normally be negotiated during the Over-The-Air (OTA) join. Sensors communicating via a satellite connected gateway will need alternate timing.

The benefit of the Rx-1 ACK is that the ACK will be on the same frequency as the original uplink message and, if the uplink used a lower SF, the ACK will also use that SF reducing the on air time. These factors will reduce the congestion due to ACK messages. The Rx-2 ACK has the benefit of using SF12 and should therefore be more resilient but if a large number of messages were ACKed using Rx-2 the 869.525MHz channel could become congested.

Class A devices that expect to receive messages from the network will need to request ACKs. Any downlink message waiting at the network server will be packaged into the ACK sent to the sensor therefore the sensor only receives the waiting message if it sends a message. If the sensor needs a response to a specific message that response may not be available until the sensor sends another message. If the sensor only transmits periodically, normally it wouldn't get a response until the next transmission. If the sensor requires a quick reply then the sensor would need to be programmed to send a message as soon as possible after the first message was sent. If the sensor has multiple messages waiting for it a flag will be set in the ACK to let the sensor know it has more messages waiting.

The process of Class-A send-and-acknowledge and send-and-reply can be seen in Fig 2. The ACK of a

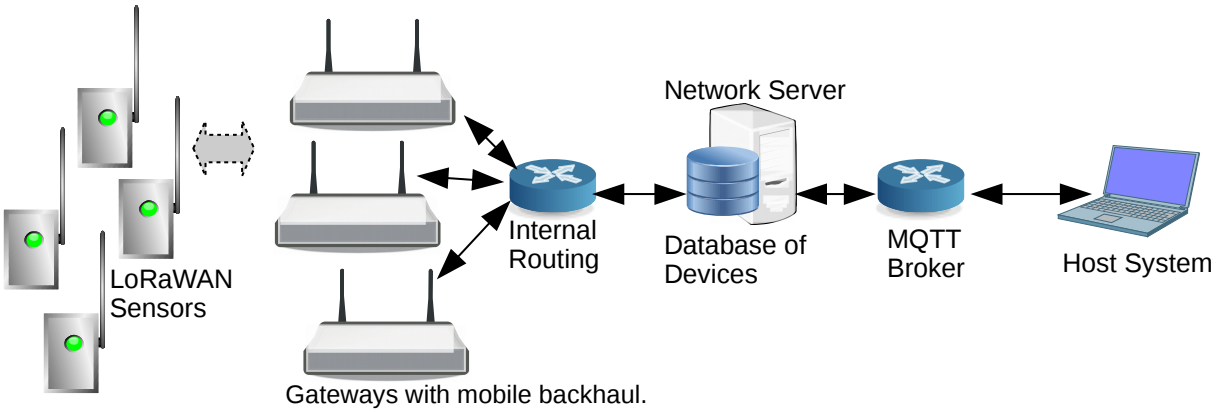

Fig.1 Generic Network Topology

message comes at a second (assuming Rx-1) but the reply to the message doesn't come until the ACK to the following message.

A. Experiment Design

This experiment was desiged to capture data on both uplink and downlink performance including the performance of the different spreading factors and the performance of the Rx-1 receive window.

The LoRaWAN device consisted of a Raspberry Pi (RPi) controlling an AT firmware Multitech mDOT with a Python script. The host system was a Javascript program running in a web page and communicating using an MQTT-over-websocket connection via a broker to the LoRaWAN Network Server.

The sensor system (RPi) operated the connection in the Class A mode with all communications initiated by the RPi. The RPi generated a monotonically increasing sequence of numbers and transmitted these via the LoRaWAN, shifting the Spreading Factor (SF) on each subsequent transmission. The LoRaWAN Network Server (NS) acknowledged these messages using the Receive Window 1 (Rx-1) , with the ACK sent via the gateway that registered the best RSSI on the uplink message. As messages were received from the RPi by the NS, these were forwarded to an MQTT Broker. The browser based client had subscribed to the channel to receive messages from the RPi. As each message was received the sequence number was extracted and echoed to the MQTT publish channel associated with the RPi. The echo message would be received by the NS and on the next incoming message from the RPi the echo message would be attached to the ACK message for transmittal to the RPi. Finally, as each ACK was received by the mDot and subsequently the RPi, any embedded message was extracted and appended to the next outgoing RPi message. This process was depicted in Fig 2.

By the method outlined above, the operation of the complete message round-trip was visible at the browser screen. Incoming messages were displayed on the screen. Outgoing messages were printed to the console. The incoming messages would include both the sequence number generated by the RPi and the echoed message generated by the browser application. As an example, the incoming message could read:

S:10237 E:10235

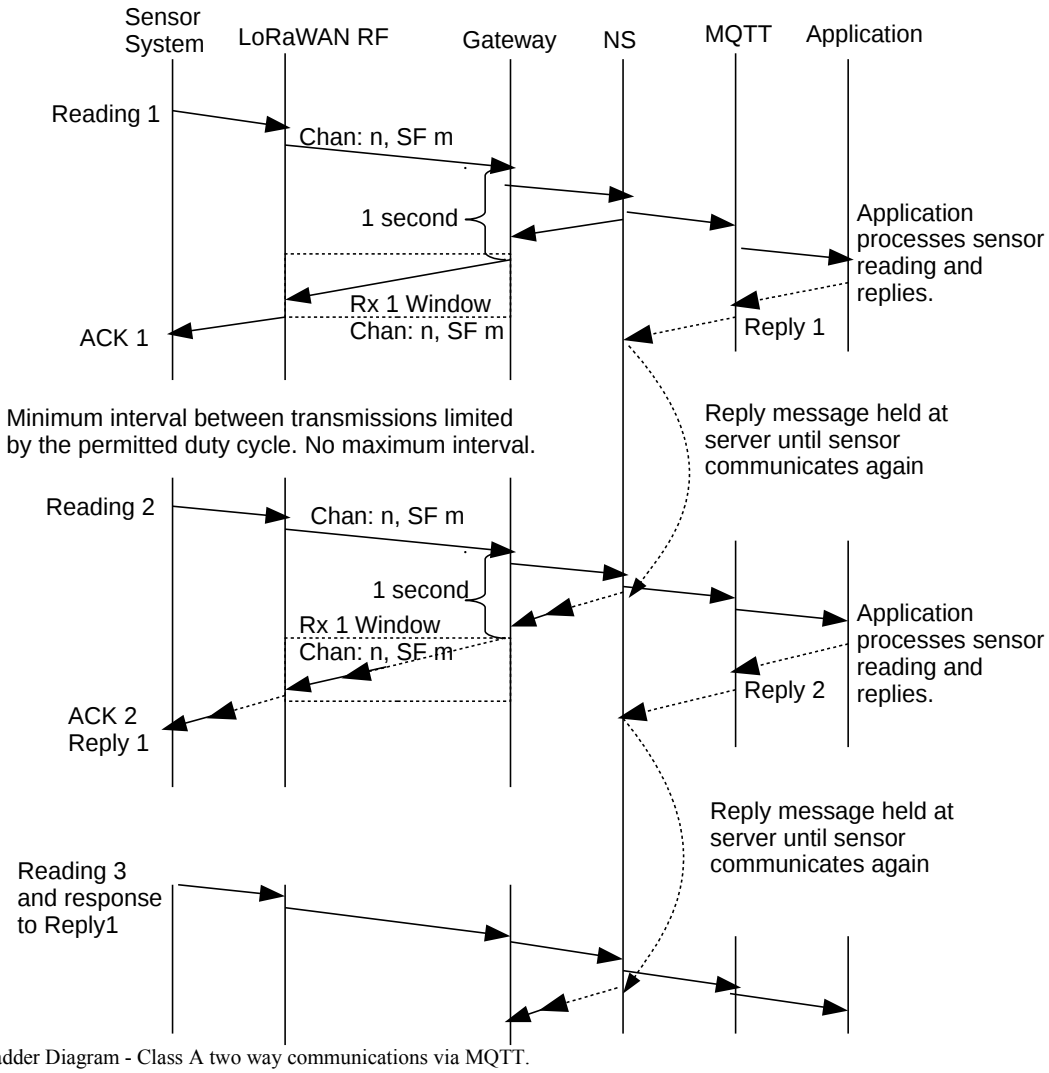

Fig.2 Ladder Diagram - Class A two way communications via MQTT.

The 'S:10237' was the sequence number generated for this message, the 'E:10235' was the echo message generated by the browser client from a previous incoming message eg: message S:10235.

B. Experiment Results

Because of the number of variables that can affect the LoRa transmission and reception there are a variety of detailed statistics presented both in tabular form and graphical form.

Table I: Summary Statistics

Uplink Messages Sent	1306
Uplink Messages Received	1297
Downlink Messages Sent	1297
Downlink Messages Received	1052
Known Downlink Errors (CRC)	31
De-duplication Errors	11

It was expected that only three gateways were 'participating' in the tests however a fourth gateway in a different reception zone began detecting messages at the higher SF. These results were included as they are informative on the network behavior at higher SF.

Table II: Test Results by Gateway and Spreading Factor

Msg Type	Msg Sent	GW 1	GW 2	GW 3	GW 4	Total Msg Rx	ACK* Rx
SF 7	217	209	154	152	0	216, 99.5%	193, 89%
SF 8	218	205	180	193	0	217, 99.5%	209, 96%
SF 9	218	211	196	200	7	218, 100%	203, 93%
SF 10	218	208	205	209	96	218, 100%	157, 72%
SF 11	218	205	207	207	111	217, 99.5%	143, 65%
SF 12	217	195	203	201	137	211, 97.2%	147, 70%
All	1306	1233	1145	1162	351	1297	1052
Rx %		94%	88%	89%	27%	99.3%	81.1%
Mean RSSI		-89 dBm	-99 dBm	-107 dBm	-112 dBm		-102 dBm
Std RSSI		2.6 dBm	3.7 dBm	4.2 dBm	0.91 dBm		7 dBm

*ACK Received percentage is percentage of the ACKs transmitted not messages transmitted.

A single gateway can successfully capture messages but multiple geographically diverse gateways that share coverage can improve the message capture rate. This table compares the number of messages captured by different pairs of gateways. The paired results from table III (below) can be compared to the results for individual gateways in table II (above).

Gateway message capture percentages as a function of SF from Table II are reproduced graphically in Fig.3. Combined with the visualiation of the ACK receive percentages (Fig.4) these results can assist in choosing the correct operational parameters for sensors.

Table III: Reception Rates for Pairs of Gateways

Gateway Pairs:	Messages Received	Percent
GW-1 + GW-2	1292	98.9%
GW-1 + GW-3	1292	98.9%
GW-1 + GW-4	1252	95.7%
GW-2 + GW-3	1235	94.6%
GW-2 + GW-4	1157	88.6%
GW-3 + GW-4	1177	90.1%

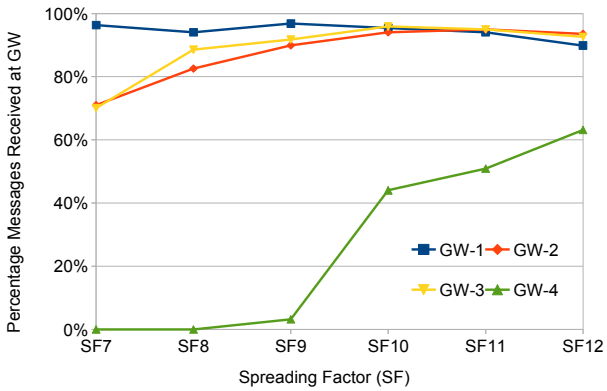

Fig.3 Gateway Reception Percentages as a Function of SF

While Fig.3 captures the performance of uplink packets at different SF in a one-to-many environment, Fig.4 captures the performance of downlink packets in a one-to-one environment.

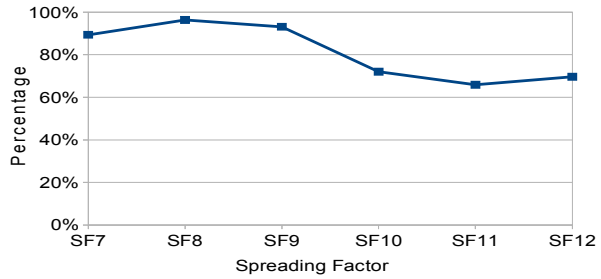

Fig.4 Percentage of ACKs Received as a Function of SF

LoRaWANs operating in Europe have three channels that all gateways should implement and these channels are used for devices attempting to join the network. Once joined the end devices can down load a channel list from the gateways to find all the available channels in that area. For this sequence only the three compulsory channels were used and the frequency vs SF for one gateway graphed below. This gateway location was chosen for the graphical results as it had previously shown an issue with a channel/SF combination and this was an opportunity to re-check. Note that although the location was the same the physical gateway was different therefore, given the results below, the issue was with the location not the gateway.

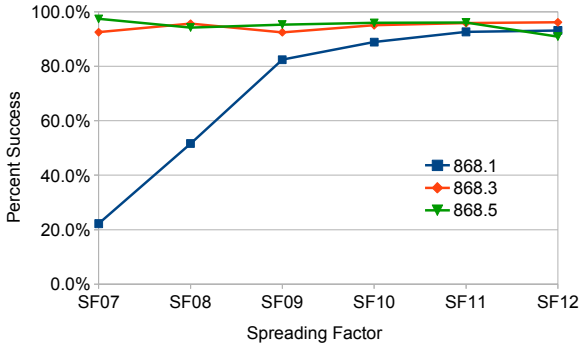

Fig.5 GW-2: Percentage of Messages Received as a Function of SF and Frequency

C. Experiment Discussion

From previously published results the high reception rate for uplink messages (99.3%) was not unexpected. The level of redundancy with three gateways capable of receiving the messages at good signal strength meant that very few messages would be missed. Despite that, in analysing the reception across the gateways even the fourth gateway contributed one message that the other three gateways had missed. For Table III it was seen that any pair of gateways performed better than a single gateway. Even GW-1, with a reception rate of 94.4% benefited in a pairing with GW-4 whose reception rate was only 26.9%. In fact GW-1 benefited more from the pairing with GW-4 (+19) than did GW-2 (+12) or GW-3 (+15). Possibly this was because GW-1 performed less well at SF12 than it did at other SF.

The reception rates for GW-4 from Table II graphed in Fig.3 clearly exhibited the improved reception due to the extra sensitivity of SF12 reception yet conversely the reception of the higher SF dropped slightly for the other Gateways. It appeared that the higher SF did not perform as well at higher receive levels and this could have been a factor in the ACK receive levels of Fig.4. The majority of the ACKs were sent from the nearest gateway with an RSSI of around -89dBm but the ACK reception at the higher SF dropped to around 70%, down from over 90% for the lower SF.

When analysing Gateway performance across SF and frequency GW-2 exhibited very poor reception on the 868.1 MHz channel for the lower SF (Fig.5). This was noted previously and it was suggested then that there was interference on that frequency. A separate test where the mDot could choose any available channel indicated that for that gateway, SF7 performed well across the spectrum except for the severe dip at 868.1 MHz.

i. Two Way Communications Results

Overall, across all SF the full round trip success was 80.6%. As noted above the performance could be affected by the chosen SF with the worst round trip success of 67.7% (at SF12) and the best round trip success of 95.9% (at SF8). This suggests that for two-way systems the host system and the sensor system should engage in a tuning exercise that selects the best settings for a particular scenario.

Close attention was paid to particular incidents in the logged communications. For example, if this test setup operated without error then the echoed sequence number should remain two numbers behind the uplink sequence number. The outgoing MQTT channel should only be holding a single message waiting to be appended to the ACK of the next incoming message.

At one point in the sequence the echoed numbers slipped to three numbers behind the uplink sequence number. This was traced to an incidence of de-duplication error with the sequence of events listed in Table IV. These events were reconstructed from the host screen data and the RPi logs.

Table IV: Effect of De-duplication Error on Outgoing Queue

Host screen text	Uplink Comment	Downlink Comment
S:10020	Msg. 10020 uplinked	Msg. 10019 in the ACK
S:10021 E:10019		Msg. 10020 in the ACK
S:10022 E:10020		Probably 10021 sent in an ACK but ACK not received.
S:10022 E:10020	duplicate of previous message due to de-duplication error.	Msg. 10022 didn't have time to be echoed by the host, empty ACK sent AND received. At this point there are now two outgoing messages in the MQTT queue (10022 and 10022)
S:10023	Msg. 10023 uplinked	ACK containing 10022 sent and lost (CRC error)
S:10024	Msg. 10024 uplinked	2nd copy of 10022 carried in the ACK
S:10025 E:10022		Msg. 10023 carried in the ACK

The outgoing MQTT queue is now holding two messages making the echo three messages behind.

Considering the mechanism of MQTT queueing and the LoRaWAN operation, if devices were only expecting a single message it would be possible for the queue to build up if more messages were inserted in the queue than were read out of it. This could present a problem if a message needed to be forwarded to the LoRaWAN end device as soon as possible but there were several messages ahead of it in the queue. If the application required this type of operation then the end device could be programmed to respond to a flag in the downlink message that informs it that there is another message waiting. In this way the end device could repeatedly poll the NS until the queue was empty.

III. Class C devices engaged in two way communications with an MQTT connected host system

The LoRaWAN Class C mode is designed to allow communication from the LoRaWAN to the sensor at anytime. For this to work the LoRa receiver at the mote must be always powered on which requires either a substantial battery, regular battery changes or connection to the mains power. Although the current draw is small (12mA for example), the continuous current draw eventually adds up. Using a 3.3V supply the power is 12mA x 3.3V or approximately 40mW. Over a year this

equates to approximately 1.3 MJoules of energy, the equivalent of the total energy stored in a 30Ah 12V lead acid battery. This sounds a lot but is considerably less than the standby power of many items of consumer electronics or the power used by a WiFi Access Point & Router.

Putting this purely as a cost against mains power, assuming a price of 10p per kWh and an 80% efficient power supply it would cost around four and a half pence per year to run. (0.00004kW / 0.8(efficiency) * 24 hours * 365 days * £ 0.10 per kWh = £ 0.0438)

A. Experiment Design

Using the Network Topology described previously, this experiment involved four Class C devices listening for messages from the network. A host system consisting of a JavaScript running in an MQTT connected browser sent a Class C message to one of the four devices every minute. Messages consisted of sequentially numbered packets with the host system cycling through each of the four devices every four minutes. The one minute message interval was to minimise the loading of the 869.525 MHz channel used for the Class C messages. A fifth Class C device was used to monitor the traffic and collect additional statistics on the network performance. This mDot was operating in full debug mode with all output logged.

The four target Class C devices consisted of four Multitech mDots running standard AT command firmware. A python script running on the controlling RPi detected the presence of the mDots and spawned a thread for each of the mDots. The thread configured the mDot, joined it to the network and send an uplink message to indicate that the mDot was online. Once this was completed the thread monitored the status of the Down Link Counter (DLC) and when a new message was received the thread read the message, logged it and transmitted the message back over the LoRaWAN to the JavaScript Host program.

Because Class C requires the network to initiate the connection this would normally be done by the network identifying the nearest gateway (as indicated by the RSSI from previous messages) and transmitting the message via that gateway. Because all the test devices were co-located the test would be limited using this mode. Instead the devices were tagged as being 'closest' to different gateways for the duration of the test, therefore giving a variety of signal strengths at the test devices.

The communications ladder diagram of Fig.6 shows the host application (right hand side) initiating the transaction with the message passing through the components before arriving at the sensor system where, in this case, the incoming message was read and converted to an outgoing message and transmitted back to the host application. For this experiment an ACK was not requested.

There were four sources of logged data for analysis including: the logs from the monitoring Class C device, the RPi thread logs, the host program screen capture and the JSON packets from the LoRaWAN NS.

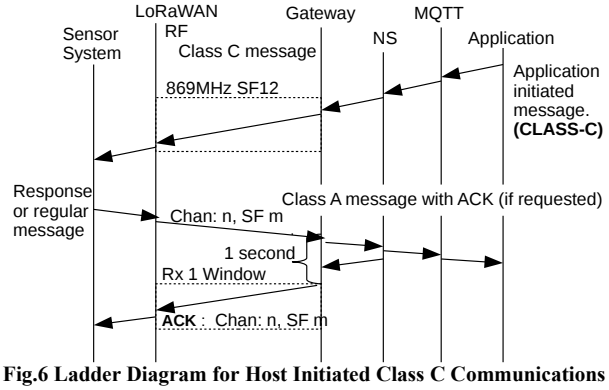

Fig.6 Ladder Diagram for Host Initiated Class C Communications

B. Experiment Results

Results for two-way Class-C communications are tabulated in Table V and Table VII (bottom of page)

Table V: Summary Statistics

Class C Messages Sent	1378
Class C Messages Received	1342 (97.4%)
3rd Party Monitor Message Received.	1340
Uplink Messages Sent	1342
Uplink Messages Received	1331 (99.2%)
Messages missed by target and 3rd party monitor	21 (of 36)

C. Experiment Discussion

The first thing to note in the results for this experiment is the discrepancy between the SF12 ACK statistics of the first experiment (Fig.4 & Tab.II) and the SF12 Class-C receive statistics for this experiment. The difference in the two experiments is that the SF12 messages in this experiment were all transmitted on 869.525MHz with no time restriction but the SF12 ACKs in the first experiment were transmitted on different frequencies and needed to be transmitted and

Table VI: Class C Results by Gateway and Receiver

Gateway	Msg Sent	mDot 1	mDot 2	mDot 3	mDot 4	mDot 5 Rx	mDot 5 %	MDot 5 mean RSSI	Common Missed*
GW 1	344		334			331	96.2%	-84.2 dBm	8
GW 2	345			341		341	98.8%	-99.7 dBm	3
GW 3	344	326				333	96.8%	-109.3 dBm	6
GW 3	345				341	335	97.1%	-109.3 dBm	4
% Success		94.8%	97.1%	98.8%	98.8%	97.2%			

*Note: 'Common Missed' were messages that both the specific mDot and the 3rd party monitor mDot missed

received in specific timeslots. The packet size should have been approximately same since both the Class-A ACK and the Class-C message were carrying contents of the same size. This suggests that there maybe some unidentified timing issue that needs further investigation.

Besides the issue identified above, the results for this test were quite good. Receive statistics for messages from the most distant gateway were marginally better than those for the nearest gateway and at 95% even the worst case is quite good. An application level protocol that sends a Class-C downlink message and requires an uplink response as an ACK wouldn't generate many resends in this particular scenario.

Results from this test were extremely encouraging.

IV. Pseudo Multicast Network

A major use-case of the LoRaWAN technology will be connection of bulk devices such as street lights. Light controllers that include LoRaWAN connectivity will still use traditional light control techniques such as sensing light levels or switching lights on at preset times but LoRaWAN connectivity will offer additional control opportunities of being able to update lighting controls singularly or in bulk. Existing radio controlled street lighting is typically a proprietry solution. The use of LoRaWAN opens up street lighting control to more service providers by decoupling the street light controller from the communications platform and the control system.

Although not necessary for lighting control, LoRaWAN multicast seems like a potential mechanism for control of lighting and other bulk services.

At the present time multicast is discussed in the LoRaWAN Specification V1.0.2 but the implementation is not fleshed out and may not be implemented in end devices. For these tests, named 'Pseudo Multicast' the devices that made up the multicast group were all set up as Class-C ABP devices with the same device address (the 4 byte network defined address) and the same network and session keys. This enabled the devices to receive and decode the broadcast multicast message. Devices would not be able to send messages while in the pseudo multicast mode since the NS uplink counters of all the different devices in the group would mean that most of the messages would be discarded by the NS. This technique doesn't stop the device also connecting under its own dev_eui as an OTAA or ABP device. A simple ad-hoc multicast group could be defined by distributing the multicast network address and keys to existing connected individual LoRaWAN devices which then store their current state (keys and counters) prior to switching over to the multicast address with associated keys and counters.

A. Experiment Design

Using the Network Topology described previously, an ABP network address with network and applications keys was created on the Stream-Technologies IoTx platfrom. The RPi controller had four Multitech mDots attached and a multithreaded Python script was written to program all the mDots as Class-C devices with the same network address and keys previously defined. Each mDot was serviced by a program thread which monitored the value of the Down Link Counter and when the counter changed, the contents of the receive buffer were read and logged. Python multi-process would have been a better solution in this scenario as the Python threading does not execute threads concurrently but the downlink message would have been received by all mDots simultaneously. This did not create an issue as there was plenty of time for all the mDots to be serviced.

The host system was a minor variation of the Class-C JavaScript Host program. For this test the host program still generated Class-C messages every minute but sent them to a single fake device address associated with the multicast network address in the NS.

As in the previous Class C experiment a fifth Multitech mDot was used as a 3rd party observer. This device was configured for Class-C working and set for full debug output with all output logged. It could not read the messages but the fact that a particular type of message was sent to a particular address was observable. The downlink counter was also visible allowing the 3rd party observer output to be matched against the outputs from individuals in the multicast group.

As previously there were four sources of logged data that could be analysed: NS forwarded JSON packets, RPi logs, Host system Screen Capture and 3rd party observer mDot communications logs.

B. Experiment Results

Results for Pseudo Multicast Network experiments are tabulated in Table VII.

Table VII: Statistics

Class C Multicast Messages Sent: 1032			
Device	Received Msg	Percent Success	Unique Loss*
mDot 1 Received	1018	98.6%	0
mDot 2 Received	1018	98.6%	0
mDot 3 Received	1016	98.4%	1+1
mDot 4 Received	1017	98.5%	1+0
mDot 5 (3rd party) Received	1016	98.4%	1+1
Specific Messages missed by all devices including 3rd party monitor			14

*Note: three mDots shared the same lost packet and two of these mDots also had a unique missed packet.

Note: The missing message unique to the 3rd party monitor appeared as a CRC Error. No other CRC error recorded by the 3rd party monitor corresponded to any missing packet.

C. Experiment Discussion

While this wasn't a true multicast experiment it was used to explore the concept and get an initial idea on how this would work in practice. Using a more distant gateway to broadcast the Class-C message might have added versimiltude although the results from the previous experiment could be extrapolated. This experiment did identify a possible issue that the previous experiment hinted at. In Table V & VI there were a number of messages missed by both the target device and the 3rd party monitor device. This could have been simple coincidence or random noise interfering at the wrong moment. The results from this multicast experiment indicate that the common missed messages were not due to coincidence. In this experiment there were 14 messages that were missed by all five listening devices. Either the messages were blanketed by noise, and this was suggested earlier in another white paper, or for some reason the network did not transmit the Class C message. The first option would be difficult to identify without additional test equipment but the second option could checked using gateway logs.

In a true multicast scenario, such as street lighting, the receivers would be geographically distributed and many receivers would be out of reach of a single gateway. In that case a practical solution would be the identification of gateways that have access to the multicast group, marking these in an administrative console and when a Class C multicast message is sent, transmitting the message from each of the tagged gateways in turn. If the message is the same then receivers that have already received the message will ignore the duplicates because the DLC is unchanged. In a similar way to that of having multiple gateways to improve the reception of uplink messages, having multiple gateways transmit the same multicast message would improve the reception rates of downlink messages.

V. Conclusions and Future Work

These experiments, while limited in some regards, continue to be encouraging in the assessment of the LoRaWAN technology. For 'set and forget' sensing where two way communication is not a requirement and where occasional missing packets are not an issue, the uplink success rate in a multi-gateway environment is very good. Many single gateway scenarios can be improved with an additional gateway that has some overlapping coverage. Two-way Class A communications had mixed results but it appeared that by choosing the appropriate operating parameters round-trip success rates of 96% were achievable. Class-C two-way communications had success rates of 95% and Class-C one-way multicast had success rates for individual receivers exceeding 98%.

Because these experiments were analysing basic packet delivery they provide useful input in developing robust application level communications protocols.

Analysing these experiments in combination has suggested some areas of further investigation. For example, why did the success rate of the Class-A Rx-1 ACK fall at higher SF when the reception of SF12 Class-C messages remained high? Some new experiments exploring this aspect will be developed and will include data gathering that can also investigate whether some messages are not processed by the network or if the missing messages are due to RF interference or other issues, for example: UDP messages to gateways lost in transit. Additional data on Gateway connection latency would also assist in analysing the performance of the network. Possibly excessive latency in mobile connected gateways leads to ACK messages being dropped.

The overall results from these experiments show that the LoRAWAN performs well in this environment. The suggested future experiments will hopefully identify some marginal issues that, once corrected, will lead to further improvements in network operation.

Appendix: De-duplication Error.

Where multiple gateways exist in a locality then multiple gateways can and will receive the same message from a LoRaWAN device. The packet forwarders on the gateways will forward the device message to the NS. Along with the message will be meta-data about the message including the network address of the device, the frequency, the SF, the RSSI and the gateway timestamp in milliseconds. When sending an ACK in the Rx-1 time-slot, the NS will add 1000ms to the gateway timestamp and tell the gateway to transmit the ACK at that time.

Where multiple gateways exist the path lengths may be different and therefore the RF packet arrival time at the gateways may be different but this difference is measured in tens of nanoseconds to tens of micro-seconds (10ns is equivalent to a path length difference of 3 metres, 20µs is equivalent to a path length difference of 6km). These times are irrelevant to the de-duplication timing. The main issue for de-duplication is the network back-haul latency particularly for mobile network connected gateways. Mobile back-haul latency could range from 200ms round trip to multiple seconds.

Consider Fig.7, messages from the LoRaWAN device are received by four gateways within a millisecond. The gateways forward the packets via their back-haul links. Three gateways have similar back-haul latency. The NS receives a packet, within a timeout period two subsequent packets match on key parameters with the first packet. A de-duplication timeout occurs and the three packets are combined into a single entity. If the uplink message included an ACK request the gateway with the best RSSI is determined, the ACK prepared and dispatched to a gateway for sending at the correct moment. A fourth packet is received but the previous reception has been dealt with and forgotten. The NS treats this as a new message and waits through a de-duplication timeout period. At the completion of this time the ACK message is prepared and sent to the gateway.

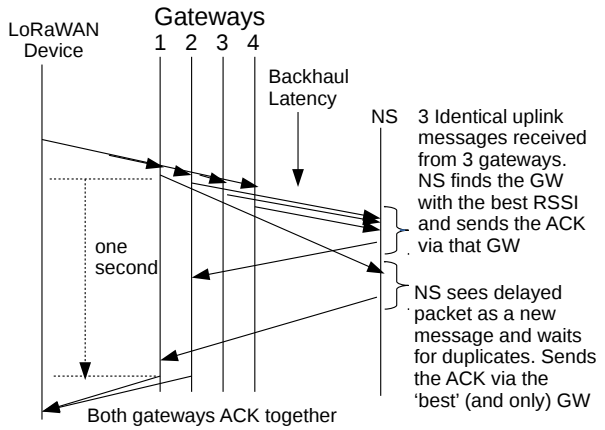

Fig.7 Communications Ladder Diagram of a De-Duplication Error

Example: given four gateways with one-way latency of 210ms, 270ms, 350ms and 400ms and a Network Server de-duplication window of 150ms:

At 210ms after the LoRaWAN message was received at a gateway the NS receives the first copy of the message including meta data. The de-duplication timing begins. By the time the 150ms has elapsed two more copies have been received. The time is now 360ms after the original LoRaWAN transmission was completed. The NS processes the packet. Assuming the 350ms gateway had the best RSSI then the ACK is returned via this gateway. Total latency and de-duplication delay accounts for 710ms. This allows plenty of processing time for the NS and the gateways.

At 400ms after the LoRaWAN message was received at a gateway and tens of milliseconds after the ACK was sent to a gateway the NS receives another copy of the message packet. This packet does not match any packet currently held in memory so the NS starts a de-duplication timer. At 550ms after the LoRaWAN packet was received the packet is processed and an ACK sent back. This ACK packet should arrive back at the gateway at approximately 950ms after the original transmission and within the one second required to set up an Rx1 ACK.

The scenario is now that two gateways will transmit, at near the exact same time an ACK on the same frequency and at the same SF. If the gateways had similar RSSI at the sensor this could result in a message collision and loss of the ACK. If there was a disparity in the RSSI then it is possible that the signal of one ACK would be sufficiently strong that the second ACK is drowned out.

An alternate scenario could be that the high latency gateway does not get the ACK message in time to set up the Rx-1 ACK packet so no ACK collision occurs.

A. De-duplication Timeout:

From the above it can be seen that the de-duplication timeout must be a tradeoff. It must be long enough that it catches most of the duplicate messages but not so long that it prevents the ACK getting back at all. Since the gateway time and date cannot be relied on and some gateway packet forwarders don't send a time and date, the NS has no knowledge of the network latency at this instant. Potentially a separate process could monitor the latency to all gateways or to a subset of gateways and this information could possibly be used to tune the NS de-duplication timing.

Glossary of Terms and Abbreviations

Term	Description
ACK	The acknowledgement message sent from the network to the end device to acknowledge a message sent from the end device to the network.
dBm	dB = decibels, a logarithmic measure. dBm is dB relative to 1mW. 10dBm is a signal power 10 times 1mW, 20dBm is a signal power 100 times 1mW. -100dBm is a sigal power 10,000,000,000 times weaker than 1mW.
De-duplication	An uplink packet can be received by multiple gateways therefore the NS will receive multiple copies of the same message, De-duplication collects all these messages together and treats it as a single message.
De-duplication Error	See the appendix on de-duplication errors for a full explanation.
DLC	Down Link Counter, count of messages sent from the Network Server
Downlink	Messages from the network to the mote
Gateway	A LoRaWAN Gateway has a sophisticated LoRa wireless concentrator for communicating with end devices and a packet forwarder that sends uplinks to the network server and received downlink packets from the NS for transmission of LoRa to the end devices.
GW	See Gateway
LoRa	Long Range Wireless.
LoRaWAN	Wide Area Network built on top of the LoRa physical layer.
mDot	A LoRaWAN development platform manufactured by Multitech. It consists of a LoRa wireless and a microprocessor with embedded LoRaWAN protocol stack.
Msg	abbreviation of Message, used interchangably with packet
Multitech	Manufacturer of communications infrastructure equipment.
NS	Network Server, the LoRaWAN connection management entity
packet	the message plus whatever protocol overhead is necesssary for the transport of the message
MQTT	Message Queue Telemetry Transport. A communications protocol that uses a message broker to receive and forward messages.
ms	milli-seconds
RSSI	Received Signal Strength Indication. A measure of the power in a received radio signal. LoRaWAN transceivers (in gateways and end devices) measure the power present in a signal and present this as the RSSI in dBm
ULC	Up Link Counter, counter of messages sent from the end device.
Uplink	Messages from the LoRaWAN sensor or mote TO the Network Server via a Gateway