MoRa: A LoRa-based System for Timely and Energy-Efficient e-Price Tag Update in Markets

Yihui wang, Gonglong Chen, Jiajun Bu and Wei Dong*
College of Computer Science, Zhejiang University.
Alibaba-Zhejiang University Joint Institute of Frontier Technologies.
Zhejiang Provincial Key Laboratory of Service Robot.
Email: {*wang1hui, desword, bjj, dongw*}@*zju.edu.cn*

***Abstract*—While numerous wireless applications are emerging to simplify our daily life, updating price tags in large-scale markets is still largely performed manually. Considering the potential adoption of electronic price (e-price) tags, we propose MoRa, a LoRa-based system for timely and energy-efficient price update in Markets. By leveraging the category information in the market, we design a hierarchical address scheme. Based on the scheme, we can perform efficient multicast with nearly zero overhead in group joining and activity scheduling. Furthermore, considering LoRa's characteristics and the asymmetry between the gateway and nodes, we let the gateway make fast and repetitive transmissions before the uncovered nodes send light weight NAKs. Extensive testbed experiments and simulation evaluations are conducted. Results demonstrate that MoRa can efficiently improve the performance in terms of update delay and energy consumption.**

***Keywords*—e-price tag, LoRa, multicast, delay, energy.**

I. Introduction

With the proliferation of low-power wireless technologies, many applications are emerging to offer efficiency and convenience for our daily life. Unfortunately, in large-scale markets where the product prices can change every two hours [1], price updating is still largely performed manually. It introduces not only intensive labor effort but also long update latency. With the emerging of low-power technologies, e-price tags show the potential to enable a wireless automatic system for price management in the market. In this paper, we consider a system for timely and reliable price updates on such e-price tags.

However, a few works have been carried out. In a recent work MarketNet [1], the authors proposed an 802.15.4-based asymmetric system for price tagging. As short-range multihop networks cause unfair energy consumption among nodes [2], they use a high-power root node to perform single-hop downlink communications. However, due to power constraints at the nodes, the uplink is still performed in a multihop fashion.

Emerging lower-power wide-area networks (LPWAN) come as promising technologies, such as NB-IoT, eMTC, etc. Amongst them, LoRaWAN has been attracting considerable interests from both academia and industry due to its open standards. Powered by batteries, LoRa nodes can reach several kilometers away for several years. LoRaWAN defines three classes to cater for different requirements in terms of delay and power consumption. Among them, providing additional downlink windows, class B is the most promising type to save power with bounded latency.

However, with nodes deciding their schedule randomly and locally, the gateway can not schedule the network resources efficiently and globally. Therefore, our design is directly based on LoRa PHY and we reschedule their activities. Thanks to its long-range, nodes are within direct reach of the gateway and network-wide synchronization can be easily achieved. However, to perform price updates for a huge number of nodes, activity scheduling can be both time- and energy- consuming. Can we avoid such unwanted overhead in the scheduling process?

In addition, considering the huge number of nodes and relatively low data rate of LoRa, the time for a network-wide update one by one can be extremely long. To reduce the delay, multicast provides a promising way by transmitting one piece of data towards a group of nodes. Generally speaking, to perform multicast, there are rounds of information exchange between nodes and the gateway for group joining and request acceptance. And for the next multicast task, the group formation may change and it will need another joining procedure. Such a joining process is again time and energy costly. Can we avoid such unwanted overhead in the joining process?

To answer the two questions above, we incorporate the category information of products in markets to design a hierarchical address scheme. Based on this scheme, joining and scheduling overhead can be largely eliminated as group information is implicitly indicated and nodes can locally schedule their reception orders. Furthermore, considering the asymmetric abilities between nodes and the gateway, we shift the most burden to the resource abundant gateway. It conducts fast and repetitive transmissions to reduce nodes' reception time and response overhead. We further reduce delays and energy consumption at the node side through light weight NAK transmissions. We incorporate these designs into a system, MoRa. It can achieve reliable multicast for e-price tag update efficiently.

The contributions of our work are summarized as follows:

- We take advantage of LoRa's long-range characteristics

*This work is supported in part by the National Science Foundation of China under Grant No. 61772465, and the Zhejiang Provincial Natural Science Foundation for Distinguished Young Scholars under Grant No. LR19F020001.

978-1-7281-2583-1/19/$31.00 ©2019 IEEE
DOI 10.1109/ICPADS47876.2019.00100

and design a market-specific hierarchical address scheme, resulting in a reduction of update delay and energy consumption by reducing joining and scheduling overhead.
- At the gateway's side, we design a fast and repetitive transmission scheme to reduce nodes' time and energy waste on data reception.
- At the nodes' side, we enable light weight NAK transmissions to cut down time and energy consumption on negotiation.
- We conduct extensive testbed experiments and simulation evaluations. Results show that MoRa can efficiently improve the performance in terms of update delay and energy consumption.

The remainder of the paper is organized as follows. Section II introduces the related work. Section III presents some preliminary studies that motivate our design. In Section IV we present the MoRa design with details. Testbed experiments and simulation evaluations are presented in Section V. Section VI concludes this paper and discusses future work.

II. Related Works

Wireless Systems for Smart Market. There are tens of thousands of products labeled with price tags in a market, of which the price can change even eight times a day [1]. Manual price update is not only labor-intensive but also time-cost. A few works have been carried out to automate this process. The electronic price label (EPL) system brought up in [3] takes the early trial. It is an RFID design which permits two-way data transfer between price tags and a controller. With a modulated backscatter uplink, it's supposed to operate on a watch battery for over five years. Recent marketNet [1] proposed an 802.15.4-based system. They adopt a high-power root to enable one-hop downlink but still perform multi-hop uplink due to range limitation. And they do not use multicast considering complex addressing and ACK explosion. Our work is the first to adopt the low-power and single-hop LoRa into the market scenario. We enable efficient multicast based on our hierarchical address design and benefit from the ACK explosion.

Wireless Multicast. In the case of IP networks, hosts use Internet Group Management Protocol (IGMP) [4] or Multicast Listener Discover (MLD) [5] to announce their interests in receiving multicast messages for IPv4 and IPv6. Router maintains a list of group members in its sub-network. Previous research efforts on multicast for Wireless Sensor Networks (WSN) mainly focus on finding a subset of nodes to relay the multicast message generated at the source node to the destinations. The majority in traditional WSN is geographic multicast algorithms. Based on the location obtained from the neighbors, multicast messages are forwarded to intended destinations [6]. These algorithms maintain a list of all destinations within the packet header, introducing overhead and impairing scalability.

With great efforts towards the seamless integration of WSNs with the Internet, standards RFC 4944 [7] and RFC 6282 [8] shaped 6LowPAN for IPv6 datagram fragmentation and header compression over 802.15.4 networks. And RFC 6550 [9] specified RPL for IPv6 routing over low-power and lossy networks (LLNs). While multicast is quite complex for IP networks, the Source-Specific Multicast (SSM) [10] simplified it for WSNs. A host joins the multicast groups with specifying source addresses it wishes to hear from. The source identification cut down the routing overhead. In the Multicast Forwarding Using Trickle [11], routers store the multicast message they have seen and they exchange such information through ICMPv6 messages at a rate controlled by Trickle [12]. Absent datagrams are forwarded subsequently. Later works further utilized tree-structure from RPL as the multicast tree and made further optimizations [13]. Our work differs in two ways. First we dopot long-range LoRa for single-hop transmission to avoid overhead on routing construction and maintenance. Second, we propose a hierarchical address design to reduce time and energy consumption on group joining.

Multicast Acknowledgement. Despite the efficiency offered by multicast, the following issue is to gather feedback from group members. Simultaneous feedbacks cause the problem of ACK storm [14]. Existing solutions fall into three categories. The first is the promiscuous reception of unicast, where multicast messages are sent to a selected receiver's unicast address while the other nodes in the multicast group listen promiscuously [15]. It achieves collision-free by RTS/CTS signaling and confine feedback to this selected node. The second is the polling-based scheme where the sender inquires about the packet reception status from each member in the multicast group sequentially. Retransmissions are made on packet loss until ACKs from all nodes are received [16]. It guarantees reliability at the cost of consuming additional network resources and can not scale. The third is leader-based schemes that a leader represents the multicast group [17]. It sends ACK upon successful reception. At the same time, other nodes will send NAK to destroy the ACK from the leader and seek retransmissions. It avoids collisions from different groups by performing RTS/CTS beforehand. Our design does not introduce additional overhead and reduces acknowledgment collection delay.

III. Motivation

In this section, we will present the most related background on LoRaWAN and LoRa, further details can be found in [18]. Then we use an example to clearly illustrate the efficiency of our design. Finally, we conduct some preliminary studies that motivate our LoRa-based design.

A. Background

In class B of LoRaWAN, periodical beacons from the gateway offer time reference for receiving nodes. As can be seen from Fig. 1, the remaining space within two beacons, the BEACON_WINDOW, is divided into 4096 ping slots of 30 ms each. Nodes decide their scheduling individually and inform the gateway of these parameters. The pingOffset indicates the random start of the first slot and the pingPeriod should be 2^k where $5 \leq k \leq 12$. Based on Chirp Spread Spectrum (CSS) modulation, LoRa has a key parameter named spreading

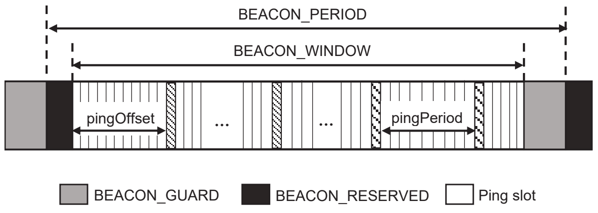


Fig. 1. Class B beacon and slot timing. PingOffset is randomly decided by individual nodes and the pingPeriod should be at least 960 ms.

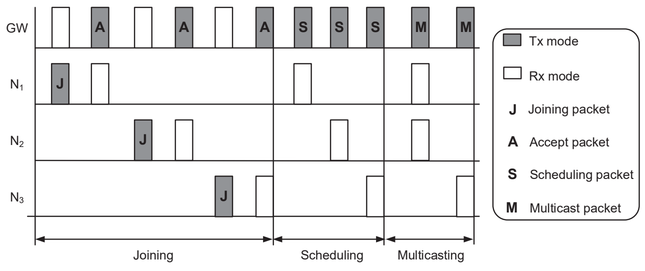


Fig. 2. A simple example of a multicast task. Each multicast task is composed of joining process, scheduling process and multicast communication. Joining and scheduling overhead take a large part of the total update delay.

factor (SF) indicating the data rate. Also, higher SF and higher transmission power are both able to boost the transmission range.

B. Example

Fig. 2 show an example of performing multicast on three nodes. N_1 and N_2 are in group one and N_3 is in group two. In every multicast task, there are joining and scheduling processes that nodes should first go through. In the joining process, a node sends the group joining message and the gateway acknowledges with necessary information like the multicast address. Then the gateway should schedule the multicast timing and ask the nodes to wake up at the corresponding time. For the sake of illustration simplicity, we assume each message an equal on-the-air time of 100 msec and there are no packet loss and no spacing between two transmissions. As can be seen from Fig. 2, it takes 1,100 msec for each multicast task where the actual multicast communications take a small portion of 18%. With the change of group formation in the next multicast task, joining and scheduling are performed again before the multicast communication. If we can avoid the joining and scheduling overhead, we can reduce the update time and energy consumption by 82%.

C. Preliminary Study

Apart from the joining and scheduling overhead, class B timing and SF value also have impacts on the update delay and energy consumption. To further understand this, we deploy 10 nodes and a gateway in our laboratory building as can be seen in Section V. To make the experiment more controllable, we let the nodes stay awake all the time and send unicast downlink data packets to them according to the class B timing. We make sure the slot interval between two nodes is long enough to accommodate traffic to and from the node. Also, the interval between the same node follows the pingPeriod regulation and

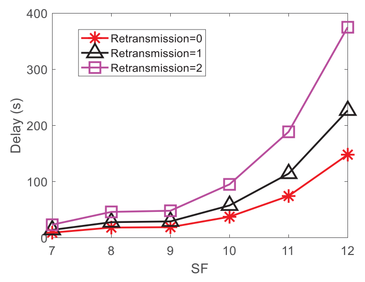
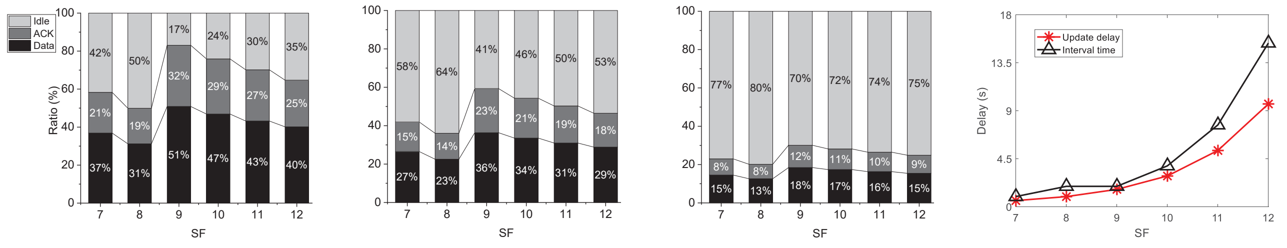


Fig. 3. Update delay of different SF under retransmissions. Delay grows exponentially with SF and a lower SF with retransmissions is much more time-efficient for guaranteeing reliability.

takes the possible smallest one. Considering 13-bytes packet header and one-bit ACK flag [18], we make the gateway transmit data packet of 23 bytes and nodes reply with 13-bytes ACK upon successful reception. We fix their transmitting power to 13 dBm and vary SF for each experiment. For every SF, the experiment is repeated for at least 5 times. We denote the update delay as the time elapses at the gateway between the first downlink data packet sent and the last uplink ACK packet received.

Insight one: Fast retransmissions. For different SF, we further classify the results into three groups according to the retransmission times at the gateway. This is due to the packet loss at a node and the gateway has to retransmit the data packet at its next assigned slot. The results are shown in Fig. 3. As expected, update delay grows exponentially with SF and adds up with retransmissions. Another important observation is that, while achieving the same reliability (100%), lower SF with retransmissions is much more time-efficient than higher SF without retransmissions. Looking from our results, the update delay of SF=12 without retransmissions can be eighteen times that of SF=7 with two retransmissions. This offers us the first insight to achieve reliability with fast retransmissions in a time-efficient way.

Insight two: LoRa PHY only. We further breakdown the update time into three parts of data transmission, ACK transmission and idle. The results are presented in Fig. 4. We can see that the idle slots contribute a lot to the total update delay, reach 50% when SF=8 even without retransmissions. It gains a larger ratio with increasing retransmissions. This is due to the sparse schedule of class B so that network resources are not optimally utilized and the update delay is unnecessarily long. We believe it would be even worse when taking into account its local decision on a random slot start. Possible slots overlapping increase the time for the gateway to address these nodes each at a time. The sudden change of ratio around SF=8 is caused by the relatively fixed slot length. Fig. 4(d) also illustrates this. To guarantee the transmission of data and ACK packets, the reception of two nodes is separated with an interval of power of 2. Slots number needed for two-way transmission of SF=8 is more than 32 thus 64 slots are assigned, which is large enough to accommodate that of SF=9. Therefore, there appears the largest idle ratio in SF=8. This gives us the second insight to base on LoRa PHY and redesign slot scheduling.



(a) Retransmission=0 (b) Retransmission=1 (c) Retransmission=2 (d) Delay and interval

Fig. 4. (a-c) Breakdown of update delay with different SF under retransmissions. There is much time wasted in idle state and the overhead of data and ACK are comparable. (d) Update delay and interval time with different SF. The update delay indicates the actual time takes for transmission and the interval time indicates the time assigned by class B specification. The gap between the two shows the inefficiency of class B scheduling.

Insight three: Ability asymmetry. Another thing we can observe from Fig. 4 is that there is comparable overhead for data and ACK transmissions. Considering the asymmetric capability of nodes and the gateway, further optimization can be made upon such a situation. For the gateway, it can transmit faster without energy constraint. For the node, the ACK overhead can be further reduced. This gives us the third insight to let the gateway transmit data packet at the possible highest power with using the fastest SF 7 and let the nodes send light-weight acknowledgments.

IV. Design

Fig. 5 shows an overview of MoRa. At the gateway's side, it receives multicast task command from the server. Based on this, the gateway decides the repetition time through scheduling module and pass it to the retransmission module. It also informs the repetition time and concerned groups to all nodes. At the node's side, the repetition time is a direct input to the slot calculation module. The concerned group addresses are input into the address matching module, through which the nodes learn their group participation. The group lists received are also an input of the slot calculation module. With an overview of the concerned groups and multicast repetition time, the node's slot calculation module outputs the wake-up timing of it. Within the scheduling slot, nodes without successful reception will perform NAK sending at the start of the collision window. At the gateway's side, it detects a successful NAK or NAK collision and conducts further retransmissions.

An example of an activity diagram is presented in Fig. 6 with a repetition time of two. N_1 and N_2 are in group one and N_3 is in group two. By receiving the broadcast message from the gateway, all nodes can compute their scheduling for multicast individually. Gateway transmits repetitive multicast messages to different groups at their corresponding wake-up time. After the multicasting phase, N_2 and N_3 go through packet losses and send NAK simultaneously at the collision window. While the two NAKs collide at the gateway, the detection of collision still asks further retransmissions from the gateway. We can see that without the joining process and with simplified scheduling, the actual multicasting forms the most portion of an update task.

Before stepping into the design details in corresponding subsections, we make clear two unique aspects of price tag

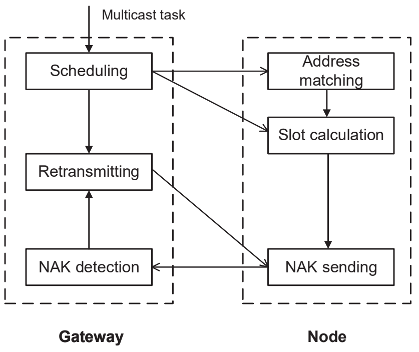


Fig. 5. Overview of MoRa.

task at markets: (1) The association between a node and a product should be throughout a node's lifetime unless manually changed. (2) For price consistency, a node should just accept price change once within a single updating task. The former gives us the confidence to assign a static address to a node once and the latter makes it safe for a node to return to sleep mode immediately after successful reception.

A. Hierarchical Multicast Address

In a large-scale market, there are tens of thousands of items. For ease of illustration, we take it as ten thousand. Therefore, market-wide reliable unicast means repeating ten thousands of two-way communications for several rounds. To cut down the number of downlink streams and thus the time budget, multicast is a promising way. However, multicast in IPv6 involves in four-way handshaking for each node in joining groups, introducing heavy communication overhead and huge energy consumption. This is also the reason why [1] uses only unicast. More generally speaking, in the traditional multicast applications, the nodes send multicast group joining messages and the gateway replies with corresponding group addresses. After nodes are all in groups, the gateway schedules the multicast timing on different groups and informs every node of this. Only after the joining process and the scheduling process will the multicast phase start. We hope to reduce or even eliminate the joining and scheduling overhead to cut down the time and energy consumption for price updating.

It is the common case that products on the racks are managed by different classifications. They are usually divided into category, sub category, sub-sub category, and type. Also,

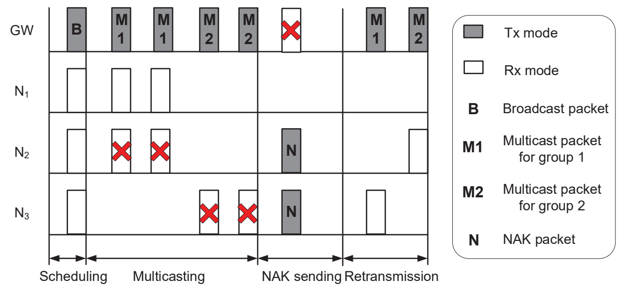


Fig. 6. An example of an activity diagram in MoRa. Nodes get essential information through the broadcast message from the gateway and locally compute their scheduling. After repetitive multicasting, N_2 and N_3 without successful reception send NAK simultaneously. NAK collision at the gateway still triggers further retransmissions.

TABLE I
AN EXAMPLE OF PRODUCTS CLASSIFICATION IN MARKETS.

Category		Sub category		Sub-sub category		Type	
1	Food	1	Frozen	1	Fruit	1	Strawberry
						2	Lemon
				2	Meat	1	Beef
						2	Pork
		2	Dry	1	Groceries	1	Vinegar
				2	Baking	1	Cookie

it is common in our daily life that all products discount for festival promotion or certain categories like vegetables and fruits are on sales considering their freshness. Table I shows a simple example. This classification of market products allows us to group nodes based on their category level. Considering the relatively static category level, group formation can remain the same for every multicast task. Therefore, we design a hierarchical address scheme based on the category information.

Take Table I for example. We simply design a four-byte hierarchical address with four fields. Suggested by the market manager, 256 is large enough to present categories at each level. Note that a complete classification list of the products from the manager will further help optimize address design in the size of each field. With each address contains all its relatively static category information, it can avoid joining varying groups for each update task. Within each field, the address of all zero is reserved. We are going to make a general elaboration on unicast, multicast and broadcast correspondingly.

- Unicast: To perform a price update on Lemon, the gateway should send a packet with the destination address of 1112H. Once upon receiving the packet, the node associated with Lemon will pass the address matching and receive the packet.
- Multicast: To perform price updating on the Fruit sub-sub category, the gateway sends a packet with the destination address of 1110H. The node represents Strawberry has an address of 1111H and the node represents Lemon has an address of 1112H. Once upon receiving the packet, they perform a process to check match from the top-level “category” field to the lowest-level “type” field. With the perfect match till “sub-sub category” field and the remaining “type” field is of all zero, they will both receive the packet. A packet with the address of 1120H or 1113H will otherwise be discarded.
- Broadcast: To perform price updating on all the products or launch commands, the gateway sends packets with a destination address of 0000H. All the nodes will find that no field match and the remaining fields are of all zero. Thus the packet will be received.

In this way, unicast, multicast and broadcast can be achieved at the nodes’ side through simply address matching. While in this paper, we only consider broadcast and multicast and leave the incorporation of unicast as future work. To reduce nodes’ idle listening and perform address matching for all incoming packets, we introduce the broadcast phase where all nodes wake up and receive the message about the repetition time and concerned groups. Through address matching, nodes learn their group in this multicast task. Then through the slot calculation, nodes get their slot scheduling and sleep until that.

B. Gateway Retransmissions

As suggested in Section III, the main-powered gateway can reach all the nodes fast at the cost of amplifying its transmitting power. To guarantee the reliability, existing works expect ACK/NAK from all nodes or some representatives soon after each transmission to perform retransmission accounting for packet loss. Considering our scenario, the gateway can transmit much faster than nodes do and the dynamic network environment is much likely to cause packet loss. Therefore, a traditional acknowledgment scheme will introduce unnecessary delay and energy consumption for the nodes to query for retransmission which is much likely to be performed.

Also, the overhead of acknowledgment can even be high and impacts the update delay. As the low-power nodes are battery-powered, SF and transmission power should be optimized to maintain connectivity. We can infer that it is much likely that nodes scattered in the market will adopt different SF and there are nodes under severe environment have to use SF=12 to keep connectivity energy-efficiently. As suggested in [18], it takes 1482.75 ms to transmit a typical 13 bytes ACK.

In this way, we defer the response from nodes after gateway’s several beforehand retransmissions. Theoretically, it takes the gateway only 61.7 ms to transmit 23 bytes message with SF=7. Also, we ease the ACK burden by redesigning a small one with only preambles.

C. NAK Response

To guarantee reliability which is much important in markets, nodes should response with gateway’s transmission to help conduct retransmissions for packet losses. Keep nodes’ relative low abilities in mind, we let nodes with successful receptions just go to sleep and nodes without receptions respond with NAK. Because NAK is the true trigger for retransmissions.

Given we perform multicast for time-efficient price updates, there are possible simultaneous NAK from the group members and cause the ACK storm problem [14]. Considering the huge number of nodes, leader-based ACK can be the promising one. In [17], the leader will transmit ACK upon successful reception or do nothing. Other nodes except for the leader will send NAK without reception. As nodes are synchronized and in the same channel, NAK from other nodes will destroy the ACK from the leader if any. Without the successful reception of ACK, the base station will schedule its retransmission. It avoids collisions from different groups by performing RTS/CTS beforehand.

However, such a scheme can not be directly adopted in our scenario. On one hand, RTS/CTS is quite a huge overhead considering the low rate of LoRa. On the other hand, transmissions with different SF are orthogonal to each other thus they operate on different virtual channels. Therefore, an ACK with SF=7 will reach gateway much faster than a NAK with SF=12 and will not be destroyed. Moreover, with nodes deciding their SF locally to combat interference and traditional CSMA not working, intra- and inter-group collisions are more severe.

In our case, we ease such burden by treating different multicast groups as a whole one. That is the gateway does not have to distinguish the received NAK from wich group and just perform multicast on all groups. This is at the expense of sacrificing a little delay. However, considering that the gateway has a rate of 20 more times than that of the nodes, straight retransmission is far more time-efficient than that after nodes' negotiation. Furthermore, as only the nodes without successful reception will stay awake for retransmissions, such a scheme will not add the energy consumption on the nodes. And we do allow nodes in the same group to waste their energy for one NAK sending, but this can be less than the consumption on rounds of negotiation and several idle listening for polling. Moreover, the maximum delay is at most the time to send a NAK with SF=12. We call it collision window which starts after the repetitive transmissions and where nodes all transmit NAK.

D. Putting Everything Together

Considering the uncertainty of price update task arrival and energy efficiency, nodes in the network may have two states, the *regular* state, and the *update* state. The state switch command is from the server-side and included in the gateway's beacon. Every node to join the network at first searches for the beacon. When the update task arrived, the nodes will switch to the *update* state. Once received the update message successfully, nodes will switch back to the regular state and only wake up for the beacon reception.

Followed by the beacon is a broadcast window. Nodes all stay awake and gateway will send a message listing the groups to participate and the number of repetitions. Nodes not indicated in groups go back to *regular* state and sleep except for beacon reception. Other nodes will compute their reception timing based on group order and repetition number. As frequent switches between sleep and awake are both time and energy consuming, transmissions before the collision window are grouped by multicast groups. That is nodes in one multicast group will stay awake for several downlink transmissions and back to sleep on successful reception or slot end. Other groups will be in sleep state until their corresponding reception timing.

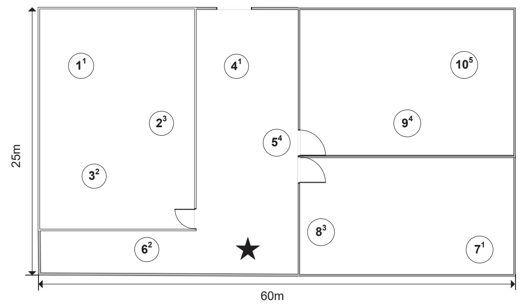


Fig. 7. Indoor setup of ten nodes (circle) and a gateway (star). The number inside each circle denotes the node ID and the superscript indicates the floor it is placed at.

In the multicast slots, nodes receive the packets from the gateway and perform the address matching. If it's addressed to it, the node performs the corresponding price update. Once a node receives the correct packet and goes through the update, it turns off its radio and switches back to the *regular* state for energy-saving.

After repetitive transmissions, the gateway expects a response from nodes in the collision window. We set its length slightly larger than the time taken for a node using SF=12 to transmit NAK. At the end of the collision window, successful NAK reception or NAK collision asks retransmission from the gateway. The length of the collision window is informed to the nodes in the broadcast slot to enable their reception for such retransmission. Otherwise, the gateway perceives the update task has been completed

V. Experiment

To verify the time and energy efficiency of our Mora design, we conduct extensive experiments with an indoor testbed. And we further conduct simulation evaluations to see the scalability with increasing nodes.

A. Testbed Evaluation

Fig. 7 presents the topology of our indoor testbed of ten Dragino LoRa Shield nodes [7]. They are deployed on different floors indicated by the superscript. We place the Dragino LG01 gateway on the roof of the same building to offer wider coverage. At both gateway and nodes, we store the log recording the reception time, sending time, SF and packet RSSI. With this testbed, we study the impact of key design parameters and exam the update delay, duty-cycle and packet reception rate. The transmission power of the nodes is set to 13 dBm. Nodes achieve time synchronization through the periodical beacons from the gateway.

SF: In the first experiment, the gateway sends packets of 23-byte payload with a transmission power of 13 dBm and varied SF from 7 to 12. The transmission interval is two seconds to generate 200 packets. The experiment is repeated for five times. As can be seen from Fig. 8(a) and 8(b), results show that while delay shows an exponential growth with increasing SF, RSSI only shows a slight improvement of 7%. At the same time, all the nodes maintain nearly 100% PRR.

Transmission power: For the second experiment, we fix

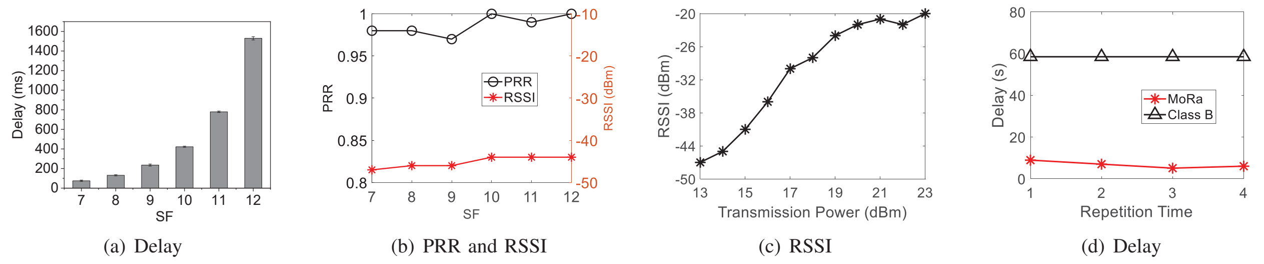


Fig. 8. Delay, PRR and RSSI with different SF. (a) (b) With PRR remaining nearly unchanged, increasing SF improves the RSSI a little at the cost of exponentially increased delay. (c) RSSI at nodes when the gateway transmits with different power. Increasing transmission power significantly improves the RSSI for larger coverage. (d) Update delay of MoRa and class B scheme.

gateway's SF to 7 and keep other settings the same as that in the previous one. We gradually increase its transmission power from 13dBm to 23dBm with a step size of 1dBm. As can be seen from Fig. 8(c), with SF=7 and transmission power of 14dBm, nodes can achieve comparable RSSI with SF=12 when the transmission power is 13dBm in Fig. 8(b). The delay reduction from SF=12 to SF=7 is 95% as can be seen from Fig. 8(a). Given that the higher RSSI the longer communication range, these results manifest the efficiency in our design to let the gateway transmit with SF=7 at a higher transmission power to save the on-the-air time.

Repetition time: We then study the repetition time with which the gateway performs fast retransmissions. MoRa incorporates such design to fully utilize the rich power at the gateway to save time for price updates and improve the reliability at first hand. The gateway transmits 23-byte packets with SF=7 at a transmission power of 23dBm, resulting in a transmission time of 61.7 msec. Nodes transmit ACK/NAK with SF=12 at a transmission power of 13dBm, resulting in a transmission time of 1,155.07 msec. Therefore, we set the collision window to 1.5 seconds. To make a fair comparison with the class B scheme, we keep the period and slot length as the same. MoRa makes several retransmissions and waits for NAK in the collision window. With class B, after every transmission, the gateway expects an ACK. A lack of it makes the gateway to retransmit to this node at its next assigned slot. The delay denotes the duration that the gateway starts beacon transmission for the price update process until the time that all nodes receive the packets. We compare the delay of MoRa and class B with varying repetition time. Both results are averaged over five experiments.

As can be seen from Fig. 8(d), MoRa gains large delay reduction over class B from 85%$\sim$92% with a repetition number from one to three. It achieves this by avoiding time wasted on querying for retransmissions all the time. Then there is a slight delay growth of 18% with repetition number of four. We infer the reason may be that nodes are all updated after three repetitions. Therefore the fourth repetition is just a waste of time. We access to nodes' log and confirm that during most experiments, the update is completed within three transmissions.

Overall performance: Finally, we present the overall performance in terms of delay and energy consumption. We design a naive approach for baseline comparison, where nodes first go through group join procedure and slot scheduling procedure. Then they receive multicast messages on their groups and response with NAK of 13 bytes soon after if needed. For MoRa, we set the repetition time to three and collision windows to 1500 msec. We perform multicast on a different number of groups each time and every experiment is repeated five times.

We obtain the overall delay as shown in Fig. 9(a). Note that, group number of ten indicates unicast and group number of one indicates broadcast. Compared to the naive approach, MoRa achieves delay reduction by 94.7% - 95.2% from broadcast to unicast. Update delay shows a clear decreased trend with the group number decreasing. This is truly the benefit from the efficiency of multicast.

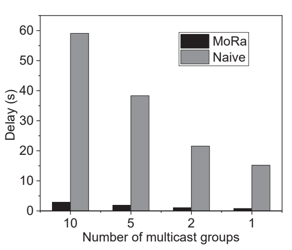
To get some further insights, we breakdown the update delay into four parts including joining, scheduling, multicasting and waiting. As the total delay of MoRa is still small compared to that of the Naive, we present their breakdown individually for a clear representation in Fig. 9(b) and Fig. 9(c) respectively. In Fig. 9(b), it is interesting to see that the time spent on multicast is the smallest portion and is overwhelmed by the constant joining and scheduling overhead and increasing waiting overhead. And in Fig. 9(c), it shows that MoRa has nearly eliminated the joining and scheduling overhead, which is 4.7$\sim$38.9 times than that is spent on multicast and NAK transmission. Also, the waiting time remains stable across different group numbers.

We also present the average wake-up time of nodes with MoRa and the Naive approach. This can be a proxy of energy consumption. As can be seen from Fig. 9(d), MoRa remains small wake-up time less than 0.5 seconds while the Naive approach has a wake-up time of more than 3 seconds.

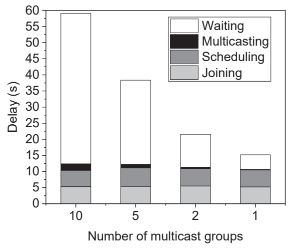
B. Simulation Studies

We further conduct simulation studies to see the scalability of MoRa compared with the naive approach. We first study the delay performance with increasing nodes number from 1000 to 10000 with a step size of 1000. Link symmetric is assumed and link quality is fixed to 90%. Then we vary the link quality from 50% to 100% with a total number of 10000 nodes. For ease of illustration, we assume four groups of nodes for multicast. All results are averaged over five-time experiments.

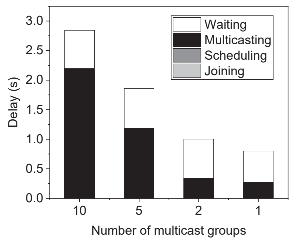
As can be seen from Fig. 10(a), MoRa maintains delay less than 0.1 hour all the time while the naive approach takes



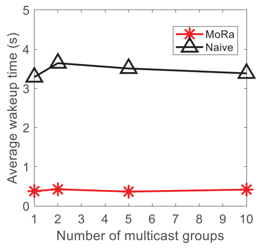
(a) Delay of MoRa and Naive



(b) Delay of Naive

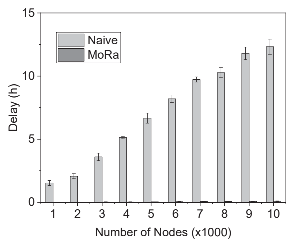


(c) Delay of MoRa

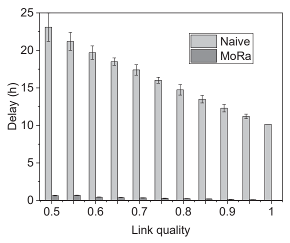


(d) Wakeup time of MoRa and Naive

Fig. 9. (a) Overall delay of MoRa and the Naive approach. (b) The delay breakdown of the Naive approach. Time spent on multicast is overwhelmed by other parts. While the time budget for joining and scheduling remain constant, the waiting time grows significantly with the group number increasing. (c) The delay breakdown of the MoRa. The overhead for joining and scheduling have been nearly eliminated. And the waiting time remains stable across different group numbers. (d) The average wake-up time of nodes with MoRa and the Naive approach.



(a)



(b)

Fig. 10. (a) Delay with varied nodes number under link quality of 90% (b) Delay with 10000 nodes under different link quality.

several hours for the same multicast. Such high latency comes from the multicast joining and scheduling procedure, and much time is wasted on waiting for NAK. In Fig. 10(b) with 10000 nodes, delay of MoRa stay below an hour even with link quality as low as 50%. It's even so small compare to that of Naive when the link quality is 100%. The simulation results manifest Mora's scalability to perform multicast updates for huge number of nodes within less than one hour.

VI. Conclusion

In this paper, we present a LoRa-based system to update e-price tags in a large-scale market both time and energy efficiently. With the category information of products, we design a hierarchical address scheme that contains all the category level information of a node on which the multicast will be performed. Based on this scheme, group joining and scheduling overhead is largelt eliminated by nodes' local computation individually. Moreover, considering LoRa's properties and the asymmetric ability between the gateway and nodes, we shift the most burden to the resource abundant gateway. Specifically, we let the gateway conduct fast retransmissions to save nodes' time on data reception. And for the nodes, we further cut down the negotiation overhead by allowing them to transmit light weight NAK directly. Extensive testbed experiments and simulation evaluations are conducted. Results show that MoRa can improve the price update performance in terms of delay and energy.

There are several directions for future work. First, we would like to explore the channel diversity for a further optimization. Second, we would like to introduce multi-gateway not only for wider coverage but also for combined stronger SNR at the nodes. It is likely to achieve faster and more reliable price update.

References

[1] H.-S. Kim, H. Cho, M.-S. Lee, J. Paek, J. Ko, and S. Bahk, "Marketnet: An asymmetric transmission power-based wireless system for managing e-price tags in markets," in *Proc. of ACM Sensys*, 2015.

[2] H.-S. Kim, Y.-J. Choi, and S. Bahk, "Elimination of multi-hop transmission from downlink in low power and lossy networks," in *Proc. of IEEE ICC*, 2014.

[3] J. G. Evans, R. A. Shober, S. A. Wilkus, and G. A. Wright, "A low-cost radio for an electronic price label system," *Bell Labs Technical Journal*, 1996.

[4] I. Kouvelas, B. Cain, B. Fenner, S. Deering, and A. Thyagarajan, "Internet group management protocol, version 3," 2002.

[5] R. Vida and L. Costa, "Multicast listener discovery version 2 (mldv2) for ipv6," Tech. Rep., 2004.

[6] D. Koutsonikolas, S. M. Das, Y. C. Hu, and I. Stojmenovic, "Hierarchical geographic multicast routing for wireless sensor networks," *Wireless networks*, vol. 16, no. 2, pp. 449–466, 2010.

[7] G. Montenegro, N. Kushalnagar, J. Hui, and D. Culler, "Transmission of ipv6 packets over ieee 802.15. 4 networks," Tech. Rep., 2007.

[8] J. Hui and P. Thubert, "Compression format for ipv6 datagrams over ieee 802.15. 4-based networks," Tech. Rep., 2011.

[9] T. Winter, P. Thubert, A. Brandt, J. Hui, R. Kelsey, P. Levis, K. Pister, R. Struik, J.-P. Vasseur, and R. Alexander, "Rpl: Ipv6 routing protocol for low-power and lossy networks," Tech. Rep., 2012.

[10] H. Holbrook and B. Cain, "Source-specific multicast for ip," Tech. Rep., 2006.

[11] J. Hui and R. Kelsey, "Multicast forwarding using trickle," *Internet Draft, Work in Progress, draft-hui-6man-trickle-mcast, January 2011*, 2011.

[12] P. Levis, T. Clausen, J. Hui, O. Gnawali, and J. Ko, "The trickle algorithm," Tech. Rep., 2011.

[13] G. Oikonomou, I. Phillips, and T. Tryfonas, "Ipv6 multicast forwarding in rpl-based wireless sensor networks," *Wireless personal communications*, vol. 73, no. 3, pp. 1089–1116, 2013.

[14] L. Rizzo and L. Vicisano, "Rmdp: an fec-based reliable multicast protocol for wireless environments," *ACM SIGMOBILE Mobile Computing and Communications Review*, vol. 2, no. 2, pp. 23–31, 1998.

[15] R. Chandra, S. Karanth, T. Moscibroda, V. Navda, J. Padhye, R. Ramjee, and L. Ravindranath, "Dircast: A practical and efficient wi-fi multicast system," in *Proc. of IEEE ICNP*, 2009.

[16] K. Piamrat, A. Ksentini, J.-M. Bonnin, and C. Viho, "Q-dram: Qoe-based dynamic rate adaptation mechanism for multicast in wireless networks," in *Proc. of IEEE GLOBECOM*, 2009.

[17] S. Choi, N. Choi, Y. Seok, T. Kwon, and Y. Choi, "Leader-based rate adaptive multicasting for wireless lans," in *Proc. of IEEE GLOBECOM*, 2007.

[18] L. A. T. Committee *et al.*, "Lorawan 1.1 specification," *LoRa Alliance, Standard*, vol. 1, p. 1, 2017.