RoCoCo: Receiver-initiated Opportunistic Data Collection and Command Multicasting for WSNs

Andreas Reinhardt[1] and Christian Renner[2]

[1] TU Clausthal, Clausthal-Zellerfeld, Germany
andreas.reinhardt@tu-clausthal.de
[2] Universität zu Lübeck, Lübeck, Germany
renner@iti.uni-luebeck.de

Abstract. Many data collection protocols have been proposed to cater for the energy-efficient flow of sensor data from distributed sources to a sink node. However, the transmission of control commands from the sink to one or only a small set of nodes in the network is generally unsupported by these protocols. Supplementary protocols for packet routing and data dissemination have been developed to this end, although their energy requirements commonly thwart the low-power nature of data collection protocols. We tackle this challenge by presenting RoCoCo in this paper. It combines data collection and dissemination by extending the low-energy ORiNoCo collection protocol by means to reconfigure subsets of nodes during runtime. Synergistically leveraging existing message types, RoCoCo allows for the definition of multicast recipient groups and forwards commands to these groups in an opportunistic fashion. Relying on Bloom filters to define the recipient addresses, RoCoCo only incurs small memory and energy overheads. We confirm its feasibility by evaluating the introduced delays, command success rates, and its energy overhead in comparison to existing collection/dissemination protocols.

1 Introduction

Wireless sensor networks (WSNs) are often used to collect environmental parameters from a range of locations at a single sink node. As the underlying embedded sensing devices are commonly confined in their available energy budget, many energy-efficient data collection protocols have been developed (e.g., [30,9,4]). These protocols are optimized for the predominant traffic type in WSNs, namely data being relayed from sensor nodes towards the sink, and thus allow for increased operational times of such networks. As collection protocols generally do not provide support for the transmission of messages from the sink to one or multiple nodes, however, changing the network configuration during runtime is complicated. If supported by the collection protocol at all, control messages (e.g., to change a node's sampling rate or to temporarily suspend its data collection) can only be flooded to all devices in the network. Besides drastically increasing the network's energy demand, having to flood each control message naïvly through the network also represents a scalability issue.

To alleviate this problem, dedicated WSN routing protocols have been devised (e.g., [29,8]). As they mostly focus on low delays while their energy consumption only plays a secondary role, however, they effectively defeat the data collection component's low-power operation when both are combined. Due to the inherent differences between data collection and control command routing, the simple combination of two such solutions is also generally suboptimal in terms of the resulting performance and energy expenditure. In order to overcome these limitations, we present RoCoCo, a sophisticated fusion of data collection and control command multicasting. Based on the low-energy ORiNoCo data collection protocol [27], it follows the primary objective of allowing for enduring operation on a tight energy budget. At the same time, RoCoCo seamlessly integrates commands and routing information into the messages used by the collection protocol in order to transfer packets from the sink to any set of nodes in the network. As a result, reconfiguration and control of individual devices becomes possible during runtime with RoCoCo at a small energy overhead. Our approach is very different from existing routing protocols like RPL [29], where a measurable amount of routing information needs to be stored at the nodes. Instead, RoCoCo relies on probabilistic data structures of constant size to make its routing decisions. We make the following contributions in this paper:

- We briefly revisit the fundamental mechanisms behind ORiNoCo and provide more detail on how destination addresses are specified by RoCoCo.
- We present RoCoCo's underlying design decisions in more detail, and highlight how it opportunistically combines energy-efficient data collection and the possibility to emit control commands to sets of nodes.
- We evaluate our solution in comparison to existing combinations of collection and dissemination protocols by means of testbed experiments as well as high-resolution power measurements.

2 Problem Statement and Background

The collection of data from distributed sensors has manifested itself as the primary application domain of WSNs. A myriad of corresponding deployments have been presented in literature, including the observation of volcanoes [28], glaciers [19], and the spreading of animal species [13]. With batteries representing the predominant energy source for the distributed nodes, however, the sensors are bound to tightly restricted energy budgets. To still achieve reasonable operational times, data collection protocols (e.g., MintRoute [30], CTP [9], Dozer [4]) were designed to forward data to the sink in an energy-efficient manner.

While catering to the transmission of data from the nodes to the sink, sending control commands from the sink to individual nodes or groups of nodes is generally beyond the capabilities of these protocols. In real applications, it may however be necessary to reconfigure a subset of nodes during runtime. Existing protocols to distribute such messages with a high probability of reaching the destination node thus often rely on broadcasting the data through the network, either by means of simple flooding or by using dissemination protocols like Trickle [16], Drip [26], or DIP [17]. None of these protocols has, however, been designed to transmit commands to a subset of nodes only, and neither have they been developed with a focus on their seamless integration with existing collection protocols. Instead, dissemination protocols are commonly orthogonal to the underlying data collection protocol, making them easily interchangeable at the price of lower energy efficiency due to the separation of components.

In this paper, we overturn this traditional separation of components by presenting RoCoCo. It combines an energy-efficient data collection protocol with means to control individual nodes or groups of nodes. We show how the fusion of functionalities can lead to a dissemination of control data at little energy overhead. Likewise, due to the use of probabilistic data structures to store the destination set for control commands, only little extra memory is required. Finally, by piggybacking control commands on messages that are sent by the data collection protocol in any case, no extra packet overhead is introduced. RoCoCo represents a novel combination of collection and dissemination protocols, yet it builds on a contribution that we have made in prior work. We thus briefly revisit ORiNoCo as follows to cater for a better understanding.

2.1 ORiNoCo: Opportunistic Data Collection

The opportunistic receiver-initiated no-overhead collection (ORiNoCo) protocol is a data-collection protocol for low-power sensor networks [27]. To achieve low power consumption, ORiNoCo duty-cycles the radio and bases its communications on low-power probing. Figure 1 illustrates a packet forwarding procedure from sender S to receiver R. Both nodes switch on their radio periodically to send short beacons that advertise their readiness to receive a packet. Each of these beacons contains a metric that models the node's path *cost* when forwarding to the sink, e.g., the hop count. If no data packet is received as response to the beacon within a short period T_{hld}, the node switches off its radio again and waits a time T_{slp}, the sleep interval, before transmitting its next beacon.

If a node S has either created a data packet itself or needs to forward data from other nodes, it switches on its radio and waits for beacon messages. Upon reception of a beacon, S decides whether to send its data packet depending on the path cost metric contained in the beacon. If the beacon's sender R offers a suitable cost, i.e., it is closer to the sink, S transmits its data packet after a small random back-off (at most T_{hld}). Successful packet reception is acknowledged by R through a beacon addressed to the data sender. Upon reception of the acknowledging beacon, S switches off its radio, if there are no more packets, or continues transmitting further packets to R. Acknowledging beacons are overheard by all nearby nodes with data to send and thus continue to serve their purpose as invitations to send data. The random back-off before packet transmissions is used to prevent collisions, should there be more than one node with

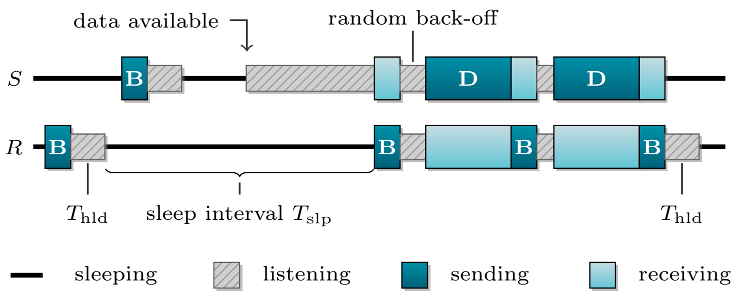


Fig. 1. Fundamental operation principle of ORiNoCo (B: beacon, D: data)

data to send. If S overhears a data packet to R (during its back-off), it aborts its sending process and waits for the next beacon. Finally, acknowledging beacons are also used to maintain the path cost metric of each node. In case the hop count metric is being used to describe the path cost, this means that a node adapts its distance to the lowest value in its vicinity plus one. To cope with link failure and changing network topology, a node resets its distance metric to a large value, if it does not receive an appropriate beacon to forward its data within a predefined time interval. In summary, ORiNoCo builds a tree-like routing structure that relies on a path cost metric to ensure that messages are only relayed towards the sink. Instead of maintaining static routes, however, each node opportunistically forwards data to any node that is closer to the sink. ORiNoCo thus has a better response to changing channel qualities and does not require nodes to maintain routing state information locally.

3 RoCoCo: Combining Opportunistic Data Collection and Control Command Multicasting

Extending a data collection protocol by means to transfer control messages from the sink to data collecting nodes poses a number of challenges. Especially as our primary objective is to retain the collection protocol's low power consumption, avoiding the energy overhead introduced by additional packet transmissions is of utmost importance. We thus present in this section how the newly introduced RoCoCo data fields are symbiotically combined with existing messages.

3.1 Destination Addressing

By convention, the sink is the final destination for all data packets in collection WSNs. Control messages, in contrast, are emitted by the sink and addressed to one or more nodes in the network. A first required step towards the distribution of a control command is thus the specification of its intended recipients. Existing protocols only support addressing a single node [29] or all nodes in the network [16,17]. We, however, argue that subsets of a WSN (e.g., nodes fitted with certain sensor types, boards of a certain hardware revision, or spatially co-located devices) may also be recipients for an emitted control command. We hence specifically design RoCoCo such that control messages can be addressed to a multicast receiver set. Depending on the number of entries, however, the set can potentially grow very large. Additionally, for WSN operating systems without dynamic memory allocation, a worst-case amount of memory must be allocated when growing lists of destination node identifiers need to be accommodated. The usage of a memory structure with constant overhead is thus inevitable to cater for the scalability to networks with a large number of nodes. Hence, we have chosen to store the set of recipient addresses for each control message in a Bloom filter [3]. The use of Bloom filters also represents the major difference to other WSN routing schemes[3], because it eliminates the need to maintain dynamically expanding lists of routable destinations.

Due to the constant memory demand of Bloom filters (BFs), the required buffers can be statically allocated, which strongly contributes to their performance on embedded systems. Because an infinite number of entries can be added to a BF, they also fulfill the requirement for message multicasting. Only the risk of false positives due to their probabilistic nature represents a downside of their usage. This potentially leads to situations in which a node may receive and execute a command although it was not among the intended recipients. As the BF is populated at the sink, where the identities of all data collecting nodes are implicitly known from the data collection, however, false positives can be detected before the command has been emitted. Details about the implementation and dimensioning of the Bloom filters used in RoCoCo are provided in Sec. 4.1, where we also analyze the introduced energy overhead.

3.2 Command Definition

Once a control message has reached its destinations, the action to take must be determined. To achieve fast reaction times, we have decided in favor of storing command identifiers within the control message. For this purpose, a field of one byte has been added to each control message, for which we have defined an initial configuration, including, e.g., commands to change the sampling rate. Please note that RoCoCo is not bound to the one byte limit for the command fields and can easily accommodate larger command definition fields. As a result of mapping the range of commands to internal functions, commands from the defined set can be immediately executed upon reception of a control message. For the transfer of larger command payloads, a reserved command identifier prompts the receiving nodes to fetch the actual command data from the sink.

3.3 Duplicate Detection

Due to the fluctuating channel qualities in WSNs, multicast control messages may reach the same node twice or even more often. However, some commands

[3] with the exception of our previous work CBFR [21] and Duquennoy et al.'s ORPL [8], neither of which however supports multicast addressing.

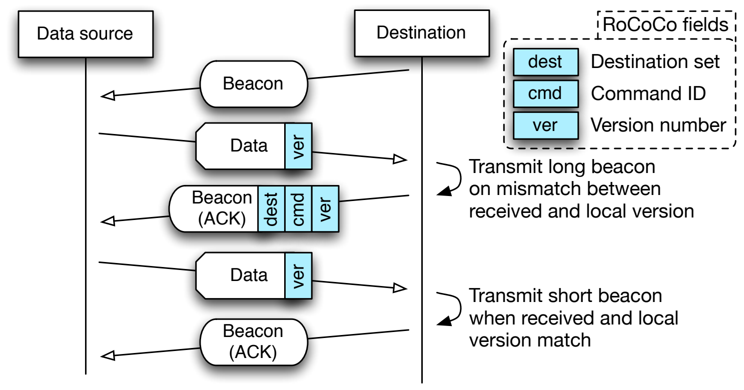


Fig. 2. Visual comparison of ORiNoCo and RoCoCo communication sequences for the transmission of 2 data packets. Fields added by RoCoCo are highlighted.

(e.g., requests to transfer a node's complete history of collected data) should only be executed once upon the first reception of the control message. In order to avoid the repeated execution of commands, we have thus incorporated a version number into the RoCoCo control messages. This two byte field is incremented whenever the sink has made any changes to at least one of the two aforementioned control fields (destination set and command identifier). As a version number thus inherently relates to a set of destination nodes and the command to execute, it can be used as a shorthand form to refer to these fields. As a result, RoCoCo uses version numbers not only as a means to avoid the duplicate execution of commands, but also to identify if a node in the network holds a newer/older command and thus needs updating. As version numbers are assigned by the sink, their consistency throughout the network is guaranteed.

3.4 RoCoCo Messages

As highlighted above, we assume a WSN in which data collection plays a major role, whereas the dissemination of control commands happens significantly less frequently. Our primary design goal has thus been to integrate the routing information fields described above into ORiNoCo in a way that retained its ultra low-power operation. As a result, RoCoCo leverages ORiNoCo's existing periodic beacon and data messages in a synergetic fashion instead of defining its own message types. As follows, we describe RoCoCo's modifications to the existing packets, that result in no additional overhead on regular beacon transmissions and only a single field added to each data packet.

Extended Beacon Messages Beacons are being sent by potential packet receivers and both serve as invitations to send as well as acknowledgments for previous data transmissions. We thus leverage them in order to disseminate control messages into the network by means of the three control message fields described above. By default, however, these optional fields are not part of transmitted beacons; beacon sizes thus are unchanged in comparison to ORiNoCo. Only when sender and receiver of a data message carry different versions of their routing information (cf. Sec. 3.5), these fields are transmitted in order to update the receiving node with the latest routing information. For the sake of clarity, we term the beacon messages that bear none of RoCoCo's newly introduced fields (destination set, command identifier, version number) as *short beacons*. We refer to beacons that contain these three entries as *long beacons*. To enable the receiver to interpret the beacons correctly, a flag was added to the previously existing ORiNoCo beacon flags field to distinguish beacon types.

Extended Data Messages Similar to the beacons, the main objective when modifying the second-most frequently used message type, i.e., data messages, was to keep the introduced overhead small. In addition to the application-defined payload, RoCoCo thus only relies on the version of the local routing information to be transmitted along with data packets. All data packets have been augmented by the version field that identifies the current routing information version of their sender. The data recipient is thus implicitly able to detect whether its communication partner has outdated routing information. If this is the case, a long acknowledgment beacon can be easily used to update the data source to the current routing version. In case both devices share the same version of the routing information, the data message is acknowledged using a short beacon.

Command Confirmation Messages Finally, we added a new message type, allowing nodes to acknowledge to the sink that they have received and executed a command. This *confirmation* message is a regular data packet and contains the control message version number.

3.5 Summary: RoCoCo vs. ORiNoCo Communication Flow

In Fig. 2, we visually compare the communication flows of ORiNoCo and RoCoCo. While ORiNoCo would only transmit the contents shown in black and white, the fields newly added by RoCoCo and required for the control command multicasting are highlighted. The operation annotation on the right-hand side represents the comparison between the received and node's local version. After the first data transmission, the destination has detected a mismatch between its local and the received version, such that the returned acknowledgment beacon is augmented by the control command data. Subsequently transmitted data packets reflect the newest version number, thus the destination only transmits short acknowledging beacons for all successive packet transmissions. Direct neighbors of the sink hence receive the updated routing information as soon as they have transmitted a packet (cf. Sec. 2.1). Still, consistent with the opportunistic nature of the data collection, we need to point out at this point that there is no guaranteed time bound for a control command to reach its destination.

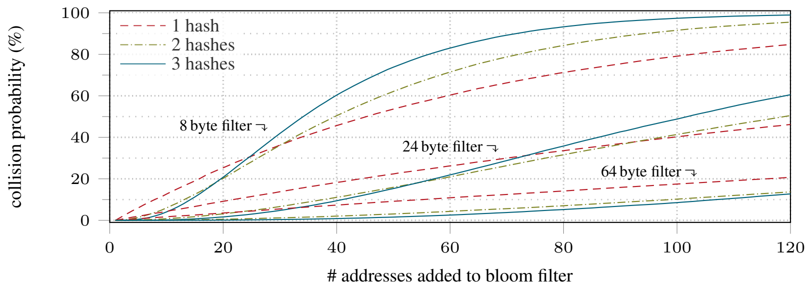


Fig. 3. Collision probability when adding a further node address to the BF

4 Evaluation

We conduct practical evaluations of RoCoCo in order to prove both its ultra low-power operation and its potential to route control messages to sets of nodes.

4.1 Bloom Filter Dimensioning and Beaconing Energy Consumption

In RoCoCo, we calculate the hash functions according to Bob Jenkins' Integer hashing[4]. In case multiple hash functions are being used, the input data is combined with index of the hash function (similar to the notion of a cryptographic salt) prior to hashing. Bob Jenkins' hash has particularly been chosen because of its speed and its minimal resource demand on motes [21]. In order to add a destination address to the BF, we take the output of each hash function modulo the size of the Bloom filter and set the resulting bit offsets in the BF.

As BFs are present in all long beacon messages, their length has an immediate impact on the protocol's energy consumption. Choosing small BF sizes thus seems desirable to minimize the energetic overhead, however, it simultaneously increases the risk of false positives. In contrast, larger BFs increase the size of long beacons and thus inherently incur a higher energy demand for their transmission. We hence analyze the tradeoff between BF size and the energy demand for its transmission. To this end, we determine the likeliness of collisions by inserting 120 sequential 16-bit node addresses into Bloom filters of 8, 24, and 64 bytes in size. In the experiment, we vary the number of hash functions from 1 to 3. The averaged collision results for 50,000 runs of the experiment with different initial addresses are shown in Fig. 3. It can be observed that the usage of a single hash function has a higher probability of collisions when a small number of addresses are inserted into the filter. At the same time, however, a larger number of hash functions leads to more collisions when more elements are inserted into the BF. A tradeoff for both the filter size and the number of hash functions thus needs to be found depending on the application's requirements.

[4] Available at `http://burtleburtle.net/bob/hash/integer.html`

Table 1. Energy demand of beacon transmissions

BF size	Energy	Overhead	BF size	Energy	Overhead	BF size	Energy	Overhead
none	155.4 µJ	reference	16 bytes	231.2 µJ	48.7 %	32 bytes	326.5 µJ	110.1 %
4 bytes	181.3 µJ	16.7 %	20 bytes	249.7 µJ	60.6 %	40 bytes	369.2 µJ	137.5 %
8 bytes	199.4 µJ	28.3 %	24 bytes	275.9 µJ	77.5 %	48 bytes	413.2 µJ	165.8 %
12 bytes	224.4 µJ	44.3 %	28 bytes	295.7 µJ	90.3 %	64 bytes	433.5 µJ	179.0 %

The number of nodes expected in the network and the permitted degree of false positives are, however, not the only criteria used for dimensioning the BF. Choosing its size also depends on the allowed energy consumption that is incurred by transmitting larger routing information packets. We have thus measured the energy demand to transmit Bloom filters of different sizes, and show the results in Table 1. All measurements were collected by means of a Hitex PowerScale [12] unit and represent the mean energy consumption as determined from three to five packet transmissions each. The table confirms that the transmission of filters has a direct impact on the energy demand over regular short beacons, with Bloom filters of 64 bytes almost tripling the beaconing energy demand.

4.2 Testbed Evaluation Setup

For all testbed experiments, we have used the following parameter set unless stated otherwise. Each node created collection data packets at an interval of 1 min. The time of the first packet was randomly chosen within the first 1 min after node reboot to simulate the behavior of an asynchronously started and operated network. Buffering queues were installed on each node with a length of 30 packets to cater for intermittent disconnection of nodes in the network due to bad radio conditions, or when incoming packets needed to be buffered before forwarding. Unless congestion occurred, collection data were thus generally sent during the next transmission opportunity. On the data collecting nodes, the mean sleep interval T_{slp} has been set to 750 ms with a random variation of 10 %. In order to achieve higher delivery, the sink has been configured to provide opportunities to receive data (by sending beacons) every 125 ms. For all nodes, we used a waiting time of $T_{\text{hld}} = 8$ ms. We employed the hop count as path cost metric and enabled the duplicate detection built into ORiNoCo.

We conducted our experiments on WiseBed (Lübeck site) [6]: This testbed is comprised of 54 TelosB nodes, numbered from 0 to 70, with each fourth address unallocated and 36 functioning nodes at the time of evaluation. Due to the size of this testbed, we have used a Bloom filter of 8 bytes in size. In all experiments, the sink issued a new command every 10 min. In order to study command and confirmation success rates and delays with an equal number of commands sent to each node, the sink always added all nodes to the BF. We conducted three experiments with different transmission powers (0 dBm, −7 dBm, and −15 dBm) to study the impact of connectivity, network depth, and density. Each experiment was run for at least 20 h, equivalent to 120 commands issued by the sink.

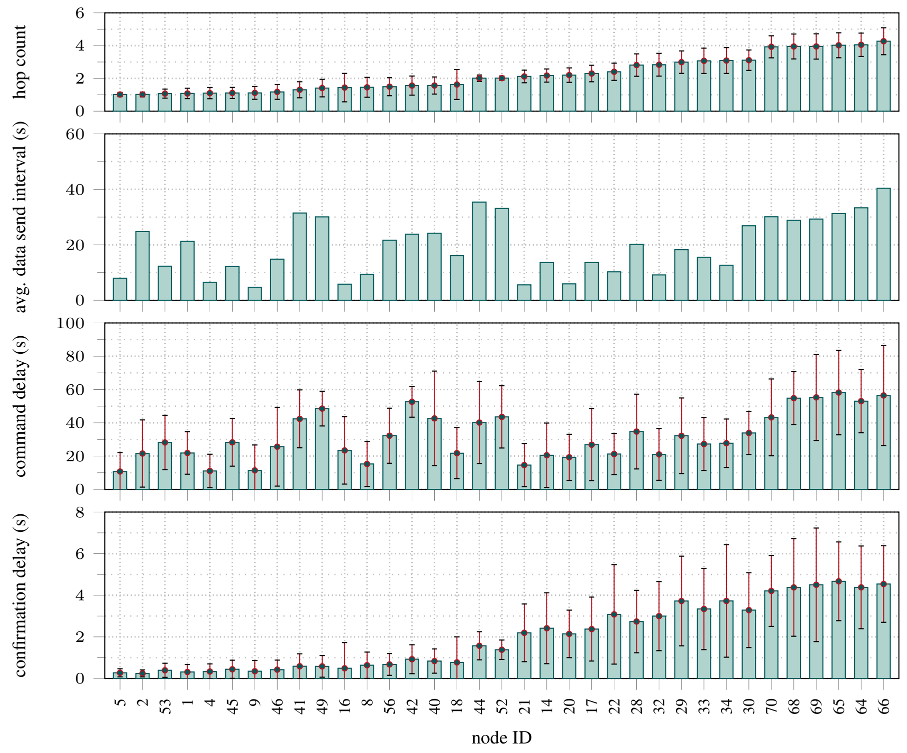
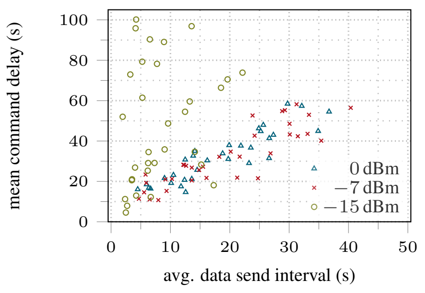


Fig. 4. Command and confirmation delay for −7 dBm transmission power. Nodes are ordered by average hop count. The figure also shows the average data (collection data and confirmations) send interval. Error bars indicate the single standard deviation.

4.3 Command and Confirmation Delays

A crucial measure for control systems is the reaction time, or command delay, of a node. By design (of RoCoCo), a node can only receive updated beacons when it has data to send and when (at least one of) its neighbors has already received the update. Therefore, the command delay to a node depends on several factors, of which the most important ones are (1) the data send interval of a node (i.e., its traffic rate), (2) its distance to the sink (in hops), and (3) the command delay to its neighboring nodes that are closer to the sink (i.e., its potential parents).

A detailed study is presented in Fig. 4. It shows the general trend that nodes with a similar data send interval (second row) exhibit a similar command delay (third row). Note that the data send interval may be smaller than 1 min (cf. Sec. 4.2), because nodes also forward (send) remote data. Data send intervals and, hence, command delay are highly topology-dependent. The figure also indicates an impact of the distance to the sink (first row) on the command delay. For nodes close to the sink, however, the data send interval is the dominating factor; e.g., compare nodes 1 and 4. However, the figure reveals exceptional behavior,



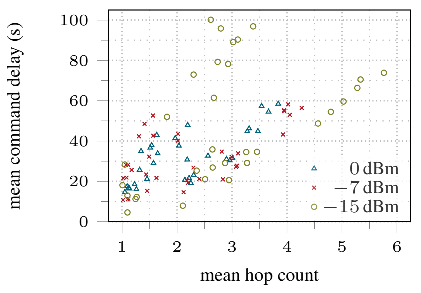


Fig. 5. Effect of network density on the relationship of command delay vs. send interval and hop count, respectively.

e.g., nodes 2 and 53 have an equal hop count and a similar command delay, yet their send intervals differ by a factor of almost two. This is due to asynchronous packet creation among all nodes (cf. Sec. 4.2) and more likely for nodes with low traffic. In contrast, the confirmation delay is mainly affected by the hop count, as confirmations are sent as regular collection data packets. Variations stem from link qualities and the number of neighbors per node. Results for the two experiments with different transmission power settings are similar.

Figure 5 analyzes the impact of network density. For dense networks (0 dBm), hop counts are lower while send intervals are longer, because shorter paths result in less traffic per node. For sparse networks (−15 dBm), hop counts are higher while send intervals are shorter, because longer paths result in more traffic per node. As a consequence, the spread of command delay is higher for sparse networks. However, some nodes (those close to the sink with a high traffic load) achieve extremely low command delays, whereas nodes with high distance to the sink are faced with longer command delays.

Next, we consider the command execution confirmation that is sent whenever a node has received a control message. As this confirmation travels in the usual direction (i.e., where all collected data flows) and represents an individual packet, its collection is considerably faster than the distribution of command messages in many cases. We assessed the round-trip time from the time when the sink issues a new command (i.e., it updates the Bloom filter) and finally receives the confirmation. For dense networks (0 dBm) the round-trip time is very close to the command delay. Due to the low hop count and the high number of potential parents, confirmations are reliably and quickly transported to the sink. On the contrary, the round-trip time may considerably deviate from the command delay in sparse networks, where long paths and few potential parents exist. This is supported by Fig. 6, which portrays the fraction of the round-trip time (RTT) caused by the confirmation. For the dense network setup, this value stayed below 14% in all cases, while it exceeded 56% in ten cases for the sparse network (not shown in the figure).

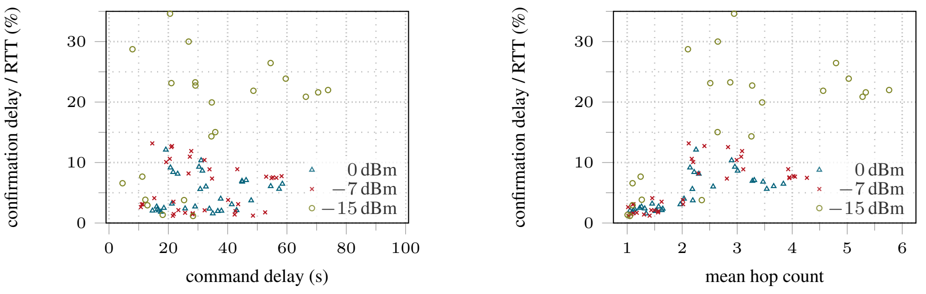


Fig. 6. Effect of network density on round-trip time vs. command delay and hop count.

Table 2. Average node power consumption for RoCoCo and existing protocols

TX power	CTP	CTP/Drip	CTP/DIP	CTP/DHV	ORiNoCo	RoCoCo
0 dBm	2.0 mW	3.9 mW	2.0 mW	4.5 mW	1.4 mW	1.4 mW
−15 dBm	1.6 mW	3.8 mW	2.0 mW	4.5 mW	1.4 mW	1.4 mW

4.4 Command Success Rates and Energy Penalty

To assess the quality of command dissemination in RoCoCo, we analyzed the success rates of command and confirmation reception. For the former, we calculated the percentage of received command messages (regardless of their destination) per node. For the testbed experiments with 0 dBm and −7 dBm, all commands and confirmations were received. For the experiment with −15 dBm, the command reception rates of all nodes range from 96% to 100% with the exception of a single node with only 90%. The percentage of command confirmations received at the sink is between 99% and 100%.

We also assessed the energy consumption penalty of RoCoCo vs. ORiNoCo by analyzing the percentile of long beacons compared to the overall number of beacons. The per-node percentile ranges from 0.1% to 1.5%. Across the entire network, the percentile of long beacons is between 0.2% and 0.4%. For a Bloom filter size of 8 bytes this equals an additional energy expenditure of less than 0.11% across the entire WSN and of less than 0.42% per node. We did not analyze the additional energy consumption incurred by the extra version field in data packets, because the impact of data packet length is minor compared to the energy consumption due to waiting for a beacon. This, however, represents the main source of energy consumption of asynchronous low-power MAC protocols.

4.5 Comparison against Existing Protocols

In order to put RoCoCo's energy demand into perspective, we have compared it against a combination of the well-known data collection protocol CTP [9] and the dissemination protocols Drip [26], DIP [17], and DHV [7]. We used the publicly available TinyOS implementations of these protocols and enabled the

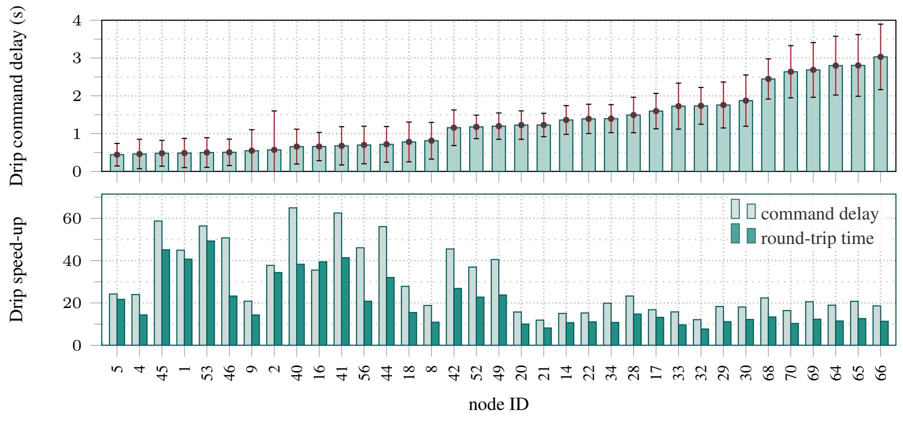


Fig. 7. Speedup of command delay and round-trip time of CTP/Drip vs. RoCoCo (−7 dBm power). Nodes are ordered by command delay (upper row) of Drip.

low-power listening MAC. The protocols were configured to use the same parameters as stated in Sec. 4.2, however with the command creation rate increased to one update per minute. The experiments were run in a one-sink one-node configuration, and the average energy consumption of the data collecting node has been practically determined for each protocol combination. The results are shown in Table 2 for transmission power settings of 0 dBm and −15 dBm. Besides highlighting that RoCoCo does not increase the energy demand of ORiNoCo measurably, the results also confirm that the RoCoCo node requires 30% less energy than the next most energy-efficient approach that combines collection and dissemination (CTP/DIP).

The combination of CTP/Drip was also run on the testbed, and statistics about the success rates plus command and confirmation delays were collected. While success rates of distributed commands are similar, CTP causes a lower success rate of confirmations; e.g., the latter ranges from 83% to 99% for the experiment with 0 dBm. It is even lower in the other experiments. Moreover, Fig. 7 shows the delays for an experiment with a transmission power of −7 dBm. In a few cases, CTP/Drip leads to a speedup factor in excess of 25 for nodes with a low command delay and close distance to the sink (primarily one- and two-hop neighbors). This speedup reduces to a factor of 10 to 25 for nodes farther away from the sink. Round-trip speedups are also around a factor of 10 for these nodes. Results for the other two experiments are similar. While the delay of command confirmations is similar between CTP/Drip and RoCoCo, command delay (and hence round-trip time) is higher for RoCoCo by design. However, note that RoCoCo has deliberately been designed to accept an increased command delay in favor of its low-power operation, while its delays (on the order of seconds) are still practical for many WSNs.

5 Related Work

Data collection in WSNs is predominantly based on static routing trees rooted at the sink (e.g., MintRoute [30]). With the introduction of opportunistic data collection, however, the restriction that each node may at most have one single parent has been removed. This possibility to choose another node for forwarding data has been shown to bear the potential for improving data throughput and reducing energy consumption [20,2]. ORW is an opportunistic data collection algorithm for sensor networks [15] and in some aspects similar to RoCoCo, although there are notable differences. While RoCoCo does not track information about its neighborhood explicitly, ORW relies on estimates such as the number of potential parents and link qualities. Most other approaches also rely on assessing link qualities, a difficult challenge in sensor networks due to the instability and low predictability of low-power wireless links. Alizai et al. hence suggest to exploit unstable, but bursty, links in [1]. Their proposed algorithm improves the performance of multi-hop routing, although an additional energy expenditure for link-quality estimation is still required.

In the domain of multicasting in WSNs, Sheth et al. presented the VLM^2 system in [23], which caters for the routing of multicast messages by maintaining stateful route information on intermediate nodes. Similarly, Chun and Tang proposed a multicasting mechanism in [5] that relies on message flooding and subsequent local matching against existing multicast group IDs. In both solutions, nodes can only subscribe to pre-defined multicast groups; a dynamic composition of multicast groups is not possible. In [24] and [22], the application of multicasting in IPv6-enabled sensor networks has been presented. The primary goal of these works is to enable nodes to join and leave IPv6 multicast groups during system operation. Neither the resulting energy demand nor the incurred routing overhead are discussed in detail, and thus their applicability in WSNs with limited energy budgets is unclear. While aforementioned approaches are primarily based on the composition of multicast groups, a number of contributions have analyzed the optimum structure of the routing tree in order to achieve delivery of messages at the smallest possible overhead ([25,31,14,10]). The papers however exclusively focus on routing when group memberships are known and provide no support for the dynamic creation and adaptation of groups.

Marchiori and Han use Bloom filters in [18] in order to route multicast messages without previous computation of the optimum route. While their PIM-WSN protocol is optimized for fast message delivery, it does not comprise a data collection component. Furthermore, it does not specifically strive for low energy consumption, rendering it inapplicable for energy-constrained data collection applications. Likewise, Heszberger et al. specify routing information by means of BFs [11], but do not combine it with a data collection protocol. Only recently, Duquennoy et al. have also presented an opportunistic point-to-point routing extension to RPL [29] in [8]. Similar to RoCoCo, their solution relies on Bloom filters to individually address nodes in the IPv6 space. In contrast to RoCoCo, however, ORPL does not support addressing a message to multiple recipients.

6 Conclusion

Due to the tightly limited energy budget of WSN nodes, ultra low-power protocols are essential to achieve long operation times. While a number of such protocols have been proposed for data collection, emitting control commands to the network is beyond their capabilities. We have thus presented RoCoCo, a lightweight extension to ultra low-power data collection that allows the sink to route control messages to sets of nodes. The fusion of data collection and control command multicasting enables administrators to configure and control the WSN during runtime. RoCoCo relies on Bloom filters to define the destination set and can thus operate in both small and large networks without any modifications. Despite its opportunistic nature, practical testbed experimentation has shown that RoCoCo achieves command dissemination success rates of 96–100% with command delays in the order of tens of seconds, even in a 36-node setting. While the observed command delays were higher than with CTP/Drip, the additional energy expenditure incurred by its application was below 0.11%. RoCoCo thus offers control command multicasting while maintaining the ultra low-power operation of the underlying ORiNoCo collection protocol.

Acknowledgments

The authors would like to thank Daniel Burgstahler, Till Schmitt, and Daniel Bimschas for their support.

References

1. Alizai, M.H., Landsiedel, O., Bitsch Link, J.A., Götz, S., Wehrle, K.: Bursty Traffic Over Bursty Links. In: SenSys (2009)
2. Biswas, S., Morris, R.: ExOR: Opportunistic Multi-Hop Routing for Wireless Networks. In: SIGCOMM. pp. 133–144 (2005)
3. Bloom, B.H.: Space/Time Trade-offs in Hash Coding with Allowable Errors. Communications of the ACM 13(7) (1970)
4. Burri, N., von Rickenbach, P., Wattenhofer, R.: Dozer: Ultra-Low Power Data Gathering in Sensor Networks. In: IPSN (2007)
5. Chun, W., Tang, W.: Multicasting in Wireless Sensor Networks. In: NGNCON (2006)
6. Coulson, G., Porter, B., Chatzigiannakis, I., Koninis, C., Fischer, S., Pfisterer, D., Bimschas, D., Braun, T., Hurni, P., Anwander, M., Wagenknecht, G., Fekete, S.P., Kröller, A., Baumgartner, T.: Flexible Experimentation in Wireless Sensor Networks. Communications of the ACM 55(1) (2012)
7. Dang, T., Bulusu, N., Feng, W.C., Park, S.: DHV: A Code Consistent Maintenance Protocol for Wireless Sensor Networks. In: EWSN (2009)
8. Duquennoy, S., Landsiedel, O., Voigt, T.: Let the Tree Bloom: Scalable Opportunistic Routing with ORPL. In: SenSys (2013)
9. Gnawali, O., Fonseca, R., Jamieson, K., Moss, D., Levis, P.: Collection Tree Protocol. In: SenSys (2009)

10. Han, K., Liu, Y., Luo, J.: Duty-Cycle-Aware Minimum-Energy Multicasting in Wireless Sensor Networks. IEEE/ACM Transactions on Networking 21(3) (2013)
11. Heszberger, Z., Tapolcai, J., Gulyas, A., Biro, J., Zahemszky, A., Ho, P.H.: Adaptive Bloom Filters for Multicast Addressing. In: HSN (2011)
12. Hitex Development Tools GmbH: Hitex PowerScale with ACM Probe, available online at `http://www.hitex.com/`, last access on 10 Sep 2014
13. Hu, W., Tran, V.N., Bulusu, N., Chou, C.T., Jha, S., Taylor, A.: The Design and Evaluation of a Hybrid Sensor Network for Cane-toad Monitoring. In: IPSN (2005)
14. Hwang, S.F., Lu, K.H., Su, Y.Y., Hsien, C.S., Dow, C.R.: Hierarchical Multicast in Wireless Sensor Networks with Mobile Sinks. Wireless Communications and Mobile Computing 12(1) (2012)
15. Landsiedel, O., Ghadimi, E., Duquennoy, S., Johansson, M.: Low Power, Low Delay: Opportunistic Routing meets Duty Cycling. In: IPSN (2012)
16. Levis, P., Patel, N., Culler, D., Shenker, S.: Trickle: A Self-Regulating Algorithm for Code Propagation and Maintenance in Wireless Sensor Networks. In: NSDI (2004)
17. Lin, K., Levis, P.: Data Discovery and Dissemination with DIP. In: IPSN (2008)
18. Marchiori, A., Han, Q.: PIM-WSN: Efficient Multicast for IPv6 Wireless Sensor Networks. In: WoWMoM (2011)
19. Martinez, K., Ong, R., Hart, J.: Glacsweb: A Sensor Network for Hostile Environments. In: SECON (2004)
20. Pasztor, B., Musolesi, M., Mascolo, C.: Opportunistic Mobile Sensor Data Collection with SCAR. In: MASS (2007)
21. Reinhardt, A., Morar, O., Santini, S., Zöller, S., Steinmetz, R.: CBFR: Bloom Filter Routing with Gradual Forgetting for Tree-structured Wireless Sensor Networks with Mobile Nodes. In: WoWMoM (2012)
22. Sá Silva, J., Camilo, T., Pinto, P., Ruivo, R., Rodrigues, A., Gaudêncio, F., Boavida, F.: Multicast and IP Multicast Support in Wireless Sensor Networks. Journal of Networks 3(3) (2008)
23. Sheth, A., Shucker, B., Han, R.: VLM^2: A Very Lightweight Mobile Multicast System for Wireless Sensor Networks. In: WCNC (2003)
24. Silva, R., Sá Silva, J., Simek, M., Boavida, F.: Why Should Multicast be Used in WSNs. In: ISWCS (2008)
25. Su, L., Ding, B., Yang, Y., Abdelzaher, T.F., Cao, G., Hou, J.C.: oCast: Optimal Multicast Routing Protocol for Wireless Sensor Networks. In: ICNP (2009)
26. Tolle, G., Culler, D.: Design of an Application-Cooperative Management System for Wireless Sensor Networks. In: EWSN (2005)
27. Unterschütz, S., Renner, C., Turau, V.: Opportunistic, Receiver-Initiated Data-Collection Protocol. In: EWSN (2012)
28. Werner-Allen, G., Johnson, J., Ruiz, M., Lees, J., Welsh, M.: Monitoring Volcanic Eruptions with a Wireless Sensor Network. In: EWSN (2005)
29. Winter, T., Thubert, P., Brandt, A., Hui, J., Kelsey, R., Levis, P., Pister, K., Struik, R., Vasseur, J., Alexander, R.: RPL: IPv6 Routing Protocol for Low-Power and Lossy Networks. RFC 6550 (2012)
30. Woo, A., Tong, T., Culler, D.: Taming the Underlying Challenges of Reliable Multihop Routing in Sensor Networks. In: SenSys (2003)
31. Wu, S., Candan, K.S.: GMP: Distributed Geographic Multicast Routing in Wireless Sensor Networks. In: ICDCS (2006)