Know when to listen

SDN-based protocols for directed IoT networks

Alves, Renan; Borges Margi, Cintia; Kuipers, Fernando A.

DOI
10.1016/j.comcom.2019.12.023

Publication date
2020

Document Version
Accepted author manuscript

Published in
Computer Communications

Know when to listen: SDN-based protocols for directed IoT networks[☆,☆☆]

Renan Cerqueira Afonso Alves[a,1,*], Cintia Borges Margi[a], Fernando A. Kuipers[b]

[a] *Universidade de São Paulo – São Paulo, Brazil*
[b] *Delft University of Technology – Delft, The Netherlands*

Abstract

Low-power wireless networks are an integral part of the Internet of Things, composed of resource-constrained devices harvesting ambient information. The appearance of unidirectional links is characteristic of low power wireless networking due to physical effects, device heterogeneity and manufacturing imperfections. Despite the prevalence of unidirectional links, most routing and radio duty cycling protocols designed for these networks do not account for such links. We provide unidirectional-link-capable protocols and study the impact of using such links on network performance indicators, such as the data delivery ratio, delay and energy consumption. Our protocols are flexible and flooding-free, leveraging centralized knowledge provided by the Software-Defined Networking paradigm. Our experiments reveal that, while unidirectional links must be detected, using them for routing enhances network performance only if the unidirectional links are long.

Keywords: Radio Duty Cycling, Software-Defined Networking, Unidirectional Links, Wireless Sensor Networks

1. Introduction

The Internet of Things (IoT) is a term used to describe the trend of inter-connecting everyday objects and sensors via the internet [3]. It spans sub-topics such as agriculture automation, smart cities, and eHealth.

Low-power wireless networks are expected to play a key role in realizing the IoT, since devices operating on batteries or harvesting energy require efficient wireless communication to save on scarce energy resources.

Homogeneous low-power wireless networks are prone to the existence of unidirectional links, which occur spontaneously due to non-isotropic antennas, multipath fading, and variations during the radio/antenna manufacturing process [30]. The occurence of unidirectional links is even higher in heterogeneous networks, due to inherent differ-

[☆] This paper is an extension of work originally presented at the 15th Wireless On-demand Network systems and Services Conference (WONS 2019) [2].

[☆☆] This study was financed in part by the Coordenação de Aperfeiçoamento de Pessoal de Nível Superior - Brazil (CAPES) - Finance Code 001.

[*] Corresponding author
Email addresses: renanalves@usp.br (Renan Cerqueira Afonso Alves), cintia@usp.br (Cintia Borges Margi), f.a.kuipers@tudelft.nl (Fernando A. Kuipers)

[1] Renan C. A. Alves is supported by grants #2016/21088-1 and #2018/11295-5, São Paulo Research Foundation (FAPESP)

ences in transmission power across device types, or through purposely increasing the transmission power of nodes plentiful in energy.

Despite the ubiquity of unidirectional links, network protocols tailored to low-power wireless communication are not aware of unidirectional links. This is in spite of the large amount of research effort into low-power routing and media access control (MAC). Some protocols blacklist unidirectional links (e.g. AODV [27]), while other protocols may not operate correctly in the presence of such links.

As far as media access control and radio duty cycling (RDC) are concerned, unidirectional links pose the challenge of operating without link-layer acknowledgements. In particular, asynchronous RDCs use acknowledgment packets to reduce the average duty cycle [11]. Bidirectional links are needed to negotiate a moment in which both sender and receiver are awake and able to communicate. For example, the classic S-MAC protocol [38] requires that neighboring nodes exchange their wake-up schedules, in addition to using a Request-To-Send/Clear-To-Send (RTS/CTS) scheme for medium access.

From the routing-layer perspective, the challenge in leveraging unidirectional links comes from efficiently computing routes that include unidirectional links. This computation is not simple; it involves disseminating the outbound neighborhood information to the sender endpoint of the unidirectional link.

The outbound neighborhood of a network node A comprises all the other nodes that can directly receive messages from A. Analogously, the inbound neighborhood of network node A is formed by all the other nodes that can directly send messages to A. Take the network depicted in Figure 1 as an example. Node 2's inbound and outbound neighborhoods are composed of the same nodes, while node 3's outbound neighborhood contains nodes 2 and 7 only, but the inbound neighborhood also includes node 6.

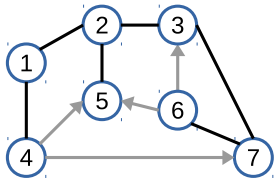


Figure 1: Network with unidirectional links. How can node 4 know it reaches 7?

On the one hand, the inbound neighbors are discovered by simply receiving messages. On the other hand, the outbound neighbors are the ones used for packet forwarding, although this information is not as easily obtained. If the link is unidirectional, the outbound neighbor needs to advertise its existence back to the sender through a multihop path. Two questions arise at this point: 1) are the unidirectional links useful; and 2) how should the reverse path to the inbound neighbor be calculated.

Considering the example in Figure 1, the link from node 4 to node 7 reduces the path length by three hops in comparison to the fully bidirectional path. Therefore, unidirectional links potentially reduce the overall energy consumption in a wireless network. But to use the 4→7 unidirectional link, node 7 needs to advertise the link existence through nodes 3, 2, and 1. How could node 7 obtain this reverse path?

In other words, our goal is to use the unidirectional links to enhance overall performance on low

power wireless networks. The problems that must be addressed to achieve this goal are producing both an efficient means of route calculation and of dissemination of reachability information, and designing a radio duty cycling algorithm able to handle unidirectional links.

We argue that centralization of network control is a simple and effective solution for using unidirectional links in low-power wireless networks. In terms of network architecture, control-plane centralization matches the Software-Defined Networking (SDN) paradigm. If network control is centralized, network nodes are exempted from discovering their outbound neighborhoods, since the centralized entity (or the controller, in SDN jargon) is responsible for calculating forwarding routes. The inbound neighborhood information is enough to enable the controller to calculate routes containing unidirectional links. Additionally, the centralization eases the wake-up phase negotiation process for radio duty cycling protocols.

This paper provides the auxiliary protocols needed for implementing the SDN paradigm in directed low-power wireless networks.

One of the protocols is the *controller discovery* protocol described in Section 2.3, whose objective is to provide a communication channel between every node in the network and the controller. This protocol is needed because there is no dedicated control channel in wireless networks (out-of-band control) as is the case in wired SDN (in-band control).

The other protocol is the neighbor discovery protocol, responsible for collecting inbound neighborhood information. Each node collects neighborhood information and sends it to the network controller. The protocol described in Section 2.2 includes mechanisms to curb overhead, to estimate link quality, and to detect neighbor departure.

We specify an asynchronous radio duty cycling protocol in Section 2.1. To the best of our knowledge, it is the first optimized RDC able to cope with unidirectional links. It uses the SDN controller to disseminate synchronization information.

Section 3 contains our experimental method, including simulation tools, protocol parameters and network topologies. The experiments were designed towards answering the question, "What are the gains from exploring unidirectional links in low-power networks?".

The results are presented in Section 4, organized by each evaluated metric, namely data delivery, link discovery rate, data delay, energy consumption and control overhead. The results show that using long-range unidirectional links is beneficial for delay, while the additional control overhead does not payoff in large topologies.

We present related work in Section 5, and our conclusions in Section 6.

2. SDN-based solution for using unidirectional links

This section describes the protocols designed to support unidirectional links, namely, the radio duty cycling, neighbor discovery and controller discovery algorithms.

2.1. Controller-aided radio duty cycling

Energy efficiency is a key performance indicator for Wireless Sensor Networks (WSN) and IoT. Since

the main source of energy consumption in such networks comes from radio communication [31], radio duty cycling (RDC) mechanisms have been devised to reduce the amount of time the transceiver is active (in transmitting or receiving states).

RDCs are classified as either synchronous or asynchronous. Synchronous protocols rely on TDMA and require a setup phase to compute the transmission schedule. Conversely, asynchronous RDCs operate without prior scheduling and do not require tight global clock synchronization.

Therefore, we focus on adapting asynchronous RDCs for unidirectional links. ContikiMAC [11] is a state-of-the-art asynchronous protocol. It uses the preamble sampling technique, in which a sender transmits a preamble with the objective of warning the receiver of the upcoming packet transmission. Receivers, in turn, periodically check the medium for ongoing transmissions.

ContikiMAC enhances the basic preamble sampling technique, reducing the preamble duration and, consequently, overall energy consumption. The main techniques employed by ContikiMAC are *preamble packetization*, *early acknowledgement* and *phase lock*.

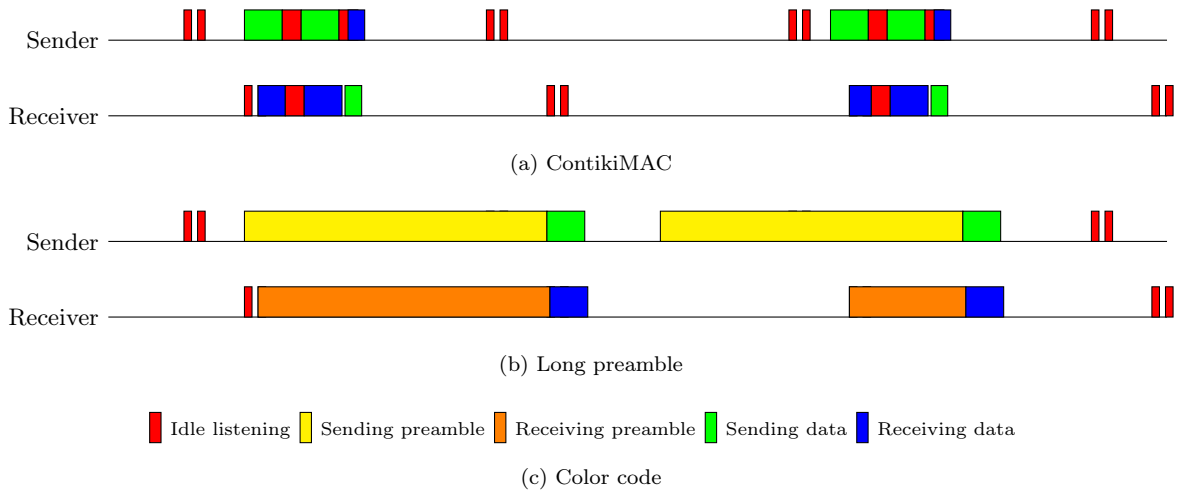
Preamble packetization consists of strobing multiple copies of the packet instead of transmitting a long preamble without useful information. Packetization enables the receiver to send an early acknowledgement, thus shortening the preamble stream. In addition, the sender can register the time at which it received the acknowledgement, calculate the phase shift between sender and receiver wake-up times and delay the start of preamble transmission.

Figure 2 exemplifies packet transmissions with two RDCs: ContikiMAC and a simple preamble sampling protocol with long preambles. The first packet transmission terminates earlier in ContikiMAC due to an early acknowledgement, while the second preamble is shorter due to the phase lock mechanism. It is noteworthy that the radio is active for a shorter amount of time in ContikiMAC.

However, ContikiMAC's improvements rely on the existence of a bidirectional link; without the receiver sending acknowledgements, the improvements do not work.

We leverage the centralized control provided by SDN to work around the lack of acknowledgements. The preamble packets are timestamped, so the receiver is able to calculate the wake-up phase shift from the sender. The receiver informs the controller about the phase shift, which, in turn, informs the sender. In the next transmission, the sender transmits only the preamble packets at the moment the receiver is expected to be listening. Figure 3a shows a transmission before the phase lock, in which the packet is strobed throughout the whole listening interval. Packet number 3 is received, and the receiver informs the controller about the calculated phase shift. In Figure 3b, the sender is already aware of the phase shift, and transmits only packet number 2, which works as a wake-up tone, and packet number 3, which is the one actually received.

The timestamp value encoded in the strobed packets is the time elapsed since the sender last woke up for medium checking. The receiver is only able to register the moment a packet is successfully received, thus the phase shift PS is calculated as $PS = T_s - (T_r + \Delta p)$, where T_s is the time stamped in the received packet (in the sender time base), T_r

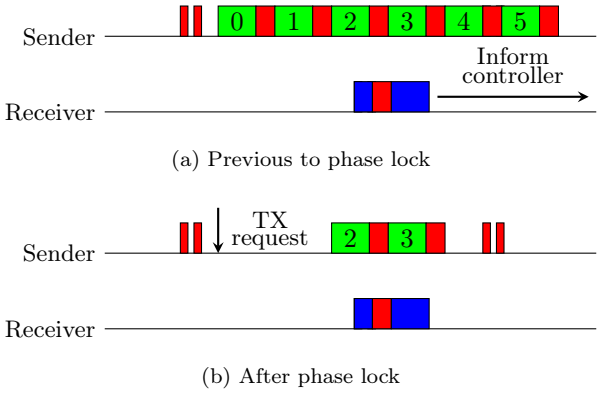


(a) ContikiMAC

(b) Long preamble

(c) Color code

Figure 2: Examples of preamble sampling techniques. ContikiMAC uses techniques to reduce energy footprint.



(a) Previous to phase lock

(b) After phase lock

Figure 3: Unidirectional RDC: the controller informs the sender when it should start preamble transmission.

is the end-of-reception time in the receiver and Δp is the packet duration in milliseconds.

In subsequent transmissions, the sender uses the phase shift information to transmit the first packet (wake-up tone) at the moment the receiver is checking the medium for transmissions. The sender transmits at time $t = PS - \frac{\Delta p + 2 * CCA + \Delta CCA}{2}$, where CCA is the time it takes to perform a clear channel assessment and ΔCCA is the time between two consecutive CCAs.

This choice of transmission time increases robustness against clock drifts, since, if the drift is zero, the receiver will wake up in the middle of the probe packet. If the receiver detects that the phase shift changed to an extent that is threatening successful packet transmissions, it should send the updated phase shift to the controller or directly to the sender. The phase shift update is triggered when the difference between the calculated value and last informed phase shift exceeds a given threshold. Complementarily, the sender could use successive phase shift updates to estimate the clock drift and automatically recalculate the current phase shift estimation, but this feature is not included in our implementation.

If the clock drift is too large, the receiver may miss packets from the sender, which would not be aware of the problem due to the lack of link-layer acknowledgements. One way to alleviate this issue is to include timestamps in broadcast packets, for which the *early acknowledgement* and *phase lock* techniques are not used. Therefore, broadcast packets can be used for resynchronization.

If too many packets are lost, the sender will eventually be removed from the receiver's neighbor table. Consequently, the controller will instruct the sender to no longer use the receiver as a relay.

Note that, in the case of communication over bidirectional links, the RDC should operate as the original ContikiMAC protocol.

2.1.1. Implementation details

This section contains implementation details and enhancements needed for the unidirectional RDC deployment.

The radio driver provides a function to load packets in the radio memory and a function to actually transmit a previously loaded packet. In the original ContikiMAC implementation, a packet is loaded to the radio only once before a sequence of strobes, while the transmit function is called repeatedly. However, the unidirectional RDC requires to load the packet before every strobe transmission, since the phase value has to be updated in the transceiver's memory.

The load function takes a non-negligible amount of time, therefore it is not feasible to load the packet immediately before the packet transmission while guaranteeing the timing constraints of the protocol. The workaround is to load the packet to the radio right after issuing a transmission. As a consequence, the packets contain the phase of the previously transmitted strobe packet, and the time calculations have to be adjusted accordingly.

The precision of the phase shift needs to be at least 0.1 milliseconds, since a 50-byte packet takes approximately 1.6 ms to be transmitted. In our implementation, we use msp430 clock ticks to measure the phase shift, which provides a precision of ≈ 0.03 ms.

The packets containing phase information are not provided with end-to-end reliability. To make sure the phase information has reached its final destination, the sender uses a bit in the packet header to inform the receiver that it knows the phase. The receiver retransmits the phase information if it receives unacknowledged unicast packets without the *known phase* bit set.

To increase robustness against clock drifts, we use the standard ContikiMAC guard time to send more strobe packets instead of only two (the wake-up tone and the received packet).

2.2. Improved neighbor discovery

The classic technique to perform Neighbor Discovery (ND) is to broadcast beacon packets at constant intervals and to assume the beacon sender is reachable by the receiver. However, due to the centralized nature of SDN, this assumption is unnecessary.

Advantages of the beacon broadcasting approach are the simplicity of the protocol and the asynchronous operation. Unfortunately, this simplicity also leads to a wasting of resources since beacon packets do not serve any other purpose and increase medium congestion.

By leveraging overhearing (Section 2.2.1) and non-constant beacon intervals (Section 2.2.2), we decrease the use of discovery packets at the expense of a small increase in complexity.

In Section 2.2.3, we describe how to integrate the task of neighbor discovery with link quality estimation. Furthermore, since discovery algorithms of-

ten focus on adding nodes in the neighbor tables and neglect node departure detection, we describe a scheme for detecting node unreachability considering unidirectional links in Section 2.2.4.

2.2.1. Neighbor discovery by overhearing

Neighbor discovery by overhearing, also known as passive neighbor discovery, is an inexpensive way of detecting surrounding nodes [37, 5]. However it may yield inconsistent discovery delays and hinder node departure detection. Therefore, we propose to jointly use passive and active discovery. The purpose of beacon packets is to advertise sender existence. However, this can be achieved by any broadcast packet, as long as the neighbor discovery protocol is informed of the reception and the addressing information is correct.

Unicast packets may also be used for discovery due to the broadcast nature of a wireless medium. However if the network uses an RDC, unicast packets are unlikely to be overheard by all neighbors, thus unicast packets cannot be used as a substitute for ND beacons.

Relying solely on overhearing increases the uncertainty of the discovery delay, as there are no guarantees of packet transmission by other protocols or applications. To overcome this drawback, a node should send periodic beacons in the absence of other packet transmissions. To achieve the desired behavior, each node sets a timer to transmit a beacon packet according to the default interval. Every time any broadcast packet is transmitted, the timer is reset, postponing the beacon transmission. This ensures a minimum packet transmission rate, guaranteeing continuous discovery while avoiding unnecessary beacons.

2.2.2. Adaptive beacon interval

Maintaining a constant beacon transmission rate is hardly the optimal strategy for saving network resources. A node should transmit more often at boot to enforce quick detection by peer nodes, while fewer packets may need to be transmitted when the network connections are stable. Therefore varying the timer interval decreases initial discovery delay and further decreases the number of discovery packets. By default, we set the initial interval to 10 seconds plus a random value based on the node id, to avoid repeated collisions. Every time a beacon is transmitted, the interval is doubled up to a maximum value (set to 2 minutes).

Using adaptive beacon intervals integrates almost seamlessly with overhearing. The only consequence being, as the transmission timer is increased when a beacon is transmitted and the overhearing mechanism avoids beacon transmission, that it may take longer to reach the maximum beacon interval.

2.2.3. Link quality estimation (LQE)

To the best of our knowledge, there has been hardly any work on link quality estimation over unidirectional links. Most packet reception ratio (PRR)-based estimators are based on acknowledged messages and calculate the metric at the transmitter (such as ETX [9], F-ETX [6], and EAR [20]). An alternative to PRR-based estimators are the hardware-based estimators, such as LQI and RSSI. However, such estimators are hardware-dependent and inaccurate [4].

ETF (Expected number of Transmissions over

Forward links) estimates the delivery at the receiver by the ratio of received probe packets over the transmitted probe packets [30]. However, implementation details are not provided, for example, how a node knows the number of transmitted probe packets, what triggers a metric calculation, and how to estimate the time window.

We provide an LQE that estimates the link quality at the receiver and does not rely on link-layer acknowledgements, as ETF. Moreover, it does not rely on probes and employs a *Moving Average* algorithm similar to Woo and Culler [36].

The receiver node maintains the status history (success or failure) of the last n messages from each inbound neighbor. However, the history is updated only when successfully receiving a message, as lost messages are not detectable.

The number of lost messages between successful receptions are calculated according to sequence numbers. A sender maintains individual sequence numbers for each outbound neighbor and for the broadcast address. Receivers calculate the number of lost packets as the difference between the current and the last received sequence number minus one.

The link quality is estimated as the number of losses over the number of entries in the history. The loss rate is preferred over the success rate to provide an additive routing metric.

The history size is a key parameter, as it directly influences the estimator reactivity, stability and granularity. Also, in the context of Software-Defined Wireless Sensor Networking (SDWSN), LQE is also responsible for triggering the ND algorithm to send a neighbor report packet to the controller due to differences between the last reported link quality estimate and the current estimate.

We have performed experiments to understand the effect of history size on the tradeoff between estimation error and reactivity. We have also studied the threshold to send updates to the controller.

Table 1 shows the mean absolute error (MAE) and the maximal error obtained from filling a history buffer from a Bernoulli distribution with the given success probabilities (p). The values are extracted from 100k samples, the MAE was extracted from all observable samples, while the maximum error was assessed only after filling the history buffer.

		Success probability (p)			
	N	0.2	0.4	0.7	0.9
MAE	8	0.131	0.144	0.131	0.084
	16	0.086	0.101	0.114	0.060
	32	0.059	0.069	0.073	0.042
Max error	8	0.675	0.600	0.575	0.650
	16	0.487	0.537	0.387	0.338
	32	0.331	0.350	0.325	0.275
Closest value	8	0.25	0.375	0.75	0.875
	16	0.25	0.375	0.6875	0.875
	32	0.1875	0.40625	0.6875	0.90625
MAE to last value	8	0.137	0.137	0.139	0.061
	16	0.083	0.098	0.091	0.047
	32	0.057	0.076	0.065	0.048
TX per 100 pkg	8	25.31	21.10	12.14	2.69
	16	11.58	10.47	5.79	1.12
	32	7.24	6.22	3.41	0.85

Table 1: Influence of history size on LQE metrics.

As expected, a larger history yields less errors: doubling the history size causes a reduction of approximately 30% in the mean absolute error. Maximum error values are not as consistent, but an overall decrease is observed.

In the context of SDWSN, the neighbor discovery algorithm is responsible for keeping the controller up-to-date with the link qualities in the network. The decision to send an update packet to the controller is based on the difference between the last reported link quality estimate and the current estimate. The controller is updated if the difference exceeds a certain threshold. A larger threshold results in less packets sent to the controller, at the cost of requiring more data to achieve a significant change in the link quality. Therefore, there is a trade-off between overhead and information freshness.

We analyze this trade-off by studying binomial distribution properties. The parameters of a binomial distribution are the success probability p and number of trials n. The distribution is defined for integer values $k \in [0, n]$. Considering $p = 50\%$, as it yields the largest variance for binomial distributions, we calculate the values of k around the distribution average ($pn = \frac{n}{2}$), such that their probabilities sum up to 80%. This calculation gives $k \in [3, 5]$ for $n = 8$, $k \in [6, 10]$ for $n = 16$ and $k \in [13, 19]$ for $n = 32$, representing a threshold of 12.5 percentage points for $n = 8$ or $n = 16$, and 9.375 percentage points for $n = 32$.

Based on these thresholds, the mean absolute error to the last transmitted value is measured as exhibited in Table 1. We observe that the error is in the same order of magnitude as the local mean error. Also in the table, we show the number of updates triggered at every 100 samples, which can be interpreted as the amount of unnecessary neighbor reports caused by statistical noise. The amount of reports is larger for low delivery probability and for smaller n.

Table 2 shows experiments regarding the reactivity by changing the success probability during the sampling experiment. The *crossing value* represents the average number of trials until the reported value is within 12.5% of the new probability, while the *first report* represents the average number of trials until a neighbor report is issued.

		Transitions			
		0.9 to	0.8 to	0.3 to	0.2 to
	N	0.8	0.9	0.2	0.3
Crossing value	8	8.5	3.6	15.2	10.8
	16	10.2	4.4	19.2	7.7
	32	6.0	6.8	14.4	7.0
First report	8	10.3	10.8	17.4	10.0
	16	20.6	23.5	29.4	16.2
	32	36.2	45.5	44.7	23.8
		0.9 to	0.2 to	0.6 to	0.4 to
		0.2	0.9	0.4	0.6
Crossing value	8	17.7	7.7	9.6	4.8
	16	25.5	13.8	13.2	7.4
	32	36.3	26.9	20.7	16.0
First report	8	7.8	2.3	8.5	5.0
	16	9.1	3.2	15.5	9.2
	32	10.8	4.2	20.4	14.4

Table 2: Reactivity study.

The crossing value is smaller when the success probability suffers little variations in comparison to when large variations occur. The reason is that the last reported value is probably already close or even already within the 12.5% range of the new probability.

If the probability changes abruptly (e.g. transition $0.2 \rightarrow 0.9$) the results show that the history buffer needs to be overwritten to achieve a

good estimation. On the other hand, the first report occurs quicker, meaning that the estimator detects a change in the probability, but avoids abrupt changes in the estimation.

Observing all values, we chose 16 as history size as the experiments indicate a balance between accuracy and reactivity.

2.2.4. Node Unreachability Detection

Detecting node departure is a tricky task as both false positives and false negatives lead to dire consequences for the established flows, causing route recalculations and decreasing the network packet delivery rate.

If a node knows it is moving or its battery is low, it could send a message advertising this information to the neighborhood (active departure detection). However, devices are usually not provided with appropriate hardware to obtain such information. Also, the cause of the link failure is often oblivious to the node, e.g., due to environmental changes. Therefore, we focus on passive node departure detection. The periodic beacon transmission is the baseline for the detection, as it sets a minimum packet transmission rate.

A receiving node knows at least one packet was lost if it has not received messages from a given neighbor for a time interval greater than the current beacon interval. As the beacon interval is not constant, the interval must be included within the packets, increasing the beacon packet size.

A neighbor is removed from the neighbor table if it fails to deliver messages for a period longer than a multiple of the beacon interval, the unreachability threshold t.

The threshold t is precalculated based on the current estimated loss rate r (provided by the LQE). Considering a maximum false negative rate of 1%, the threshold t is calculated as the t such that $r^t < 1\%$. The value of t is limited to a minimum of 2, to avoid false positives, and to a maximum of 8, to avoid extending the departure detection. Threshold values and the corresponding false negative rate are shown in Table 3 considering all possibilities of a 16-bit estimation.

Loss rate estimate	Threshold	False negative rate
0.00%	2	0.00%
6.25%	2	0.39%
12.50%	3	0.20%
18.75%	3	0.66%
25.00%	4	0.39%
31.25%	4	0.95%
37.50%	5	0.74%
43.75%	6	0.70%
50.00%	7	0.78%
56.25%	8	1.00%
62.50%	8	2.33%
68.75%	8	4.99%
75.00%	8	10.01%
81.25%	8	18.99%
87.50%	8	34.36%
93.75%	8	59.67%

Table 3: Loss threshold to remove a neighbor from the neighbor table.

2.3. Improved controller discovery

The controller discovery problem is inherent to SDWSN and requires a global algorithm to solve the general unidirectional link case. Particularly in unidirectional circle topologies, such as illustrated

in Figure 4a, a global algorithm is required. For example, the only way for node 7 to know it reaches the controller directly is for that information to propagate via node 1 throughout node 6.

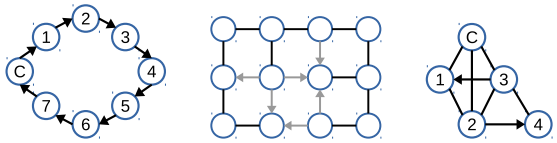


Figure 4: a) Unidirectional circle. b) Network with unidirectional links and a bidirectionally connected component. c) Example network.

If the network graph contains a bidirectionally connected component, that is, the topology graph is still connected if all unidirectional links are removed, as the example in Figure 4b, then controller discovery can be solved by a local algorithm. We believe it takes very specific radio and environmental conditions to result in a pure unidirectional network. Therefore, the existence of a bidirectionally connected component is plausible in practice.

With this assumption in mind, building a tree rooted at the controller provides an efficient solution to the problem, although nodes out of the bidirectionally connected component are not able to join the network. Since the route set by the controller discovery is temporary, the algorithm does not seek the optimal route in terms of link quality and relies on hop count to build the tree.

The controller is initialized with the minimum hop count value, while the other nodes are initialized with the maximum value. The controller discovery packets contain the current hop count towards the controller and the set of known inbound neighbors (obtained from the neighbor discovery protocol), so the receivers can check if the link is symmetric.

Each node checks its neighbor table size at exponentially increasing intervals (up to a maximum value) and transmits controller discovery packets if the number of neighbors increased. Upon receiving a controller discovery packet, the node checks if there is a bidirectional link to the sender and if the hop count is better than the current value. If both conditions are true, the next hop towards the controller and the hop count are updated and the SDN layer is informed of the discovery. Also, a controller discovery packet is scheduled for transmission, regardless of neighborhood changes.

A node N_1 also transmits a controller discovery packet whenever it already knows how to reach the controller and receives a controller discovery packet with the maximum metric from node N_2. This condition indicates N_2 still does not know a next hop towards the controller and N_1 is a next hop candidate. This procedure is intended to speed up the discovery by late nodes and allow new nodes to join the network after the initial bootstrap.

Although an individual node does not know whether the other nodes have already obtained a valid controller route, the controller discovery algorithm eventually stops sending messages if the network topology is stable and the nodes' neighborhoods remain constant.

Take the network of Figure 4c as an example. First, the controller detects nodes 1, 2, and 3 as inbound neighbors and transmits a controller discovery beacon with this information. As the links are bidirectional, these nodes can reach the controller directly. In the next round, nodes 1, 2, and

3 transmit their own beacon with neighborhood information. Node 2 does not switch the next hop to 1 or 3, because it is a longer route. Node 4 sets the next hop as node 3, since the beacon received from node 2 does not contain its address, and transmits a beacon to advertise its discovery. At this point, controller discovery beacons are no longer transmitted as 1) the topology is stable and the set of neighbors does not change, and 2) none of the nodes will change their next hop towards the controller.

3. Evaluation method

We designed experiments towards answering the question, "What are the gains from exploring unidirectional links in low-power wireless networks?". To this end, we test combinations of discovery algorithms and radio duty cycling algorithms under several scenarios with and without unidirectional links.

Firstly, we analyze a set of simulations using a pure CSMA/CA medium access scheme (i.e., without duty cycling), aiming to assess the impact of exploring unidirectional links without the extra overhead imposed by the RDC. We compare a traditional discovery algorithm used in the literature, namely the Collect-based discovery [23], to two versions of our discovery algorithms: 1) using unidirectional links, and 2) blacklisting unidirectional links. This set of experiments is displayed in Section 4.1.

A second set of experiments studies the influence of radio duty cycling on exploring unidirectional links. We deployed two versions of our RDC protocol: 1) phase information is always sent to the controller, and 2) phase information is directly sent to the target node. As a comparison, we instruct the controller to calculate only bidirectional paths and use the original ContikiMAC RDC protocol. In either case, we used a channel check rate of $8Hz$. As a baseline, we also present the outcome of the Collect-based discovery protocol combined with ContikiMAC. Experiments with RDC are discussed in Section 4.2.

Table 4 summarizes the combinations of algorithms used in the performance evaluation.

Discovery algorithm	RDC
Our – using unidir links	CSMA/CA
Our – bidir links only	CSMA/CA
Collect-based	CSMA/CA
Our – send to controller	Our
Our – send to node	Our
Our – bidir links only	ContikiMAC
Collect-based	ContikiMAC

Table 4: Combinations of algorithms tested.

In addition to fully bidirectional networks, we perform experiments in three unidirectional settings: 1) random unidirectional links (15% of all links), which emulate unidirectional links that naturally emerge in homogeneous networks, 2) random nodes with increased range (20% of all nodes have double range), to represent heterogeneous networks, and 3) a special case for SDN, referred to as "controller to all" in Section 4, in which the controller is able to reach all nodes in the network within one hop.

The number of nodes ranges from 16 to 100 nodes (square numbers only) to check the algorithms behavior as the network gets larger. The nodes are

positioned to form square grids and random topologies. The controller is positioned at a grid corner, while the data sink is placed at the grid midpoint. The positioning of these nodes is random in the random topologies. All nodes in the network transmit CBR data, except the data sink and the controller node. The data payload size is 10 bytes, transmitted at 1 packet per minute.

For each parameter combination, ten 60-minute-long simulation runs are executed to achieve statistical significance. The graphs presented in the following section show 95% confidence intervals. Every "statistically equal" or "statistically different" statement in the results description (Section 4) corresponds to the outcome of a two-tailed Mann-Whitney U-test considering $\alpha = 0.05$.

A summary of the simulation parameters is presented in Table 5.

Simulation parameters	
Number of nodes	16, 25, 36, 49, 64, 81, 100,
Topologies	random, grid
Simulation duration	3600 s
Number of replications	10
Data payload size	10 bytes
Data transmission rate	1 packet/min
Node boot interval	$[0, 1]$ s
Data traffic start time	$[2, 3]$ min
ContikiMAC channel check rate	8 Hz

Energy consumption parameters	
Radio module transmission power	0 dBm
Transmission current consumption	21.70 mA
Receiving current consumption	22.00 mA
Sleeping current consumption	0.18 mA
Operation voltage	3 V

Table 5: Simulation parameters.

3.1. Tools

All protocols are implemented on Contiki OS. The SDN support for wireless networks is provided by IT-SDN [23], an SDWSN framework and southbound protocol that allows for changing of the discovery algorithms. IT-SDN is configured to use end-to-end acknowledgements, source-routed control packets, and neighbor table size of 10 entries. The code used in the experiments is available for download at `http://www.larc.usp.br/users/cbmargi/www/it-sdn/it-sdn_comcom.tar.gz`.

The algorithms are benchmarked with the COOJA WSN simulator/emulator tool [26], using sky mote binaries and Directed Graph Radio Medium (DGRM) to model the radio links. DGRM enables defining unidirectional links, opposed to the other available radio medium models. Each link is individually defined as an ideal communication channel.

The controller software runs on the host machine and connects with the network through the COOJA serial server extension. It calculates the best routes according to link quality values provided by the neighbor discovery protocol.

The random topologies are generated with the NPART software [25], using the default parameters for Berlin networks.

3.2. Metrics

We have considered the following performance metrics:

- *Data delivery*: the global percentage of data packets that successfully reached their destination.

- *Data delay*: the average time between the data packets transmission and reception (at the application layer, therefore queuing and flow setup delays are included).

- *Control overhead*: the total number of non-data packets transmitted within the network, which is related to the discovery algorithm's efficiency.

- *Energy consumption*: the total amount of energy spent by the radio transceiver of all network nodes, considering three radio states (transmitting, receiving, and off). We have used Energest [12] (a tool from Contiki OS) to obtain the amount of time spent in each state. The energy consumption is calculated as $E = V(I_tT_t + I_rT_r + I_iT_i)$, where V, T, and I are the voltage, time spent in each state, and current drawn in each stage; the subscripts t, r and i refer to transmitting, receiving and idle states, respectively. Drained current values have been taken from the CC2420 datasheet [34] and replicated in Table 5.

- *Link discovery rate*: the percentage of existing links that the neighbor discovery algorithm was able to detect throughout the simulation. The discovery rate is measured at the controller, considering its global representation.

4. Results and discussion

The results are organized as follows: each metric is presented as a set of four graphs, one for each link type (i.e. fully bidirectional, controller to all, nodes with increased range, and random unidirectional links). Within the graphs, each line represents a combination of three simulation parameters: 1) neighbor discovery algorithm (i.e. collect or this work); 2) radio duty cycling protocol (i.e. pure CSMA, ContikiMAC, or this work); and 3) topology type (i.e. random or grid).

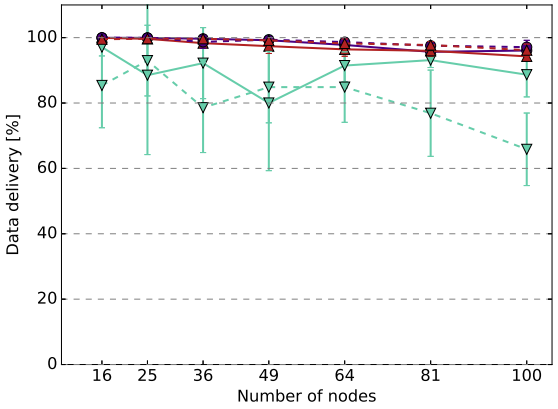
4.1. Pure CSMA results

This section contains experiment results for non-duty-cycled pure CSMA networks, performed with the objective of assessing the impact of using unidirectional links for routing without the influence of radio duty cycling. For brevity, we include only data delivery and data delay results.

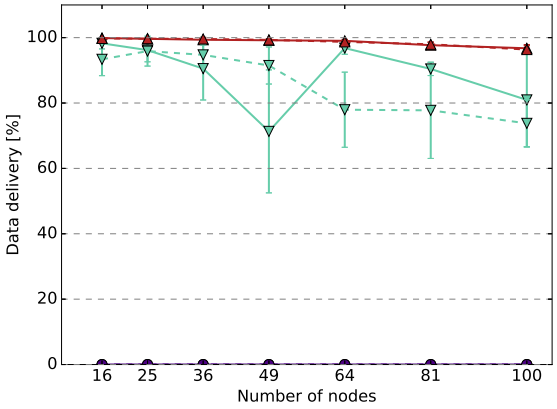
Figure 5 displays data delivery results. In fully bidirectional networks, our discovery algorithms perform as good as the Collect-based approach, presenting statistically equivalent results in most network sizes (Figure 5b). However, our algorithms perform worse if unidirectional links are blacklisted (between 5% and 30% worse), although this is only statistically significant in the grid topology. The reason behind these results is that each direction of the link is informed separately, creating temporary unidirectional links in the controller representation that cannot be used for routing.

In fact, we can observe an increased packet delivery rate when using unidirectional links in the other scenarios. Enabling the use of unidirectional links yields at least 90% delivery, while not using these links can drop the delivery into the realm of 80%. We can notice a larger impact in the "controller to all" and "random unidirectional links" topologies. The delivery rate is statistically different when us-

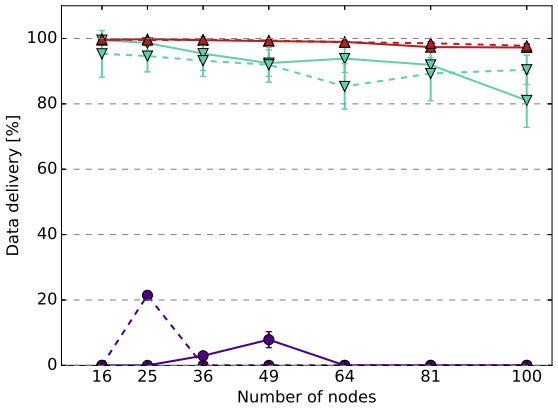
(a) Legend



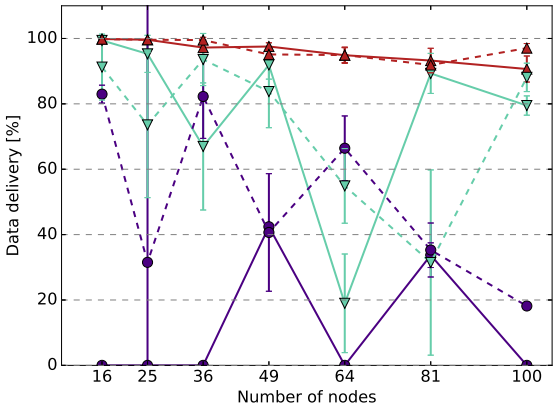
(b) Full



(c) Controller to all



(d) Random nodes with increased range



(e) Random unidirectional links

Figure 5: Pure CSMA: Data delivery results.

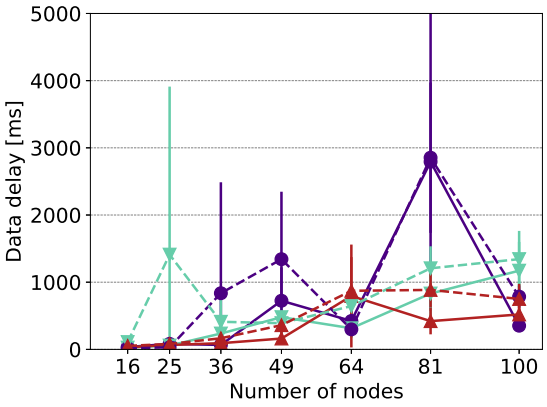
ing and not using unidirectional links for all scenarios with unidirectional links, except for 16-node and 81-node random topologies with random unidirectional links.

It is notable that some curves do not present a monotonic delivery decrease as the number of nodes increases, especially in Figure 5e. The reason behind that is the randomness in topology and unidirectional link placement. Particularly, these two factors may be combined in such a way that the remaining bidirectional links become bottlenecks. As such, as the results show, using unidirectional links increases robustness.

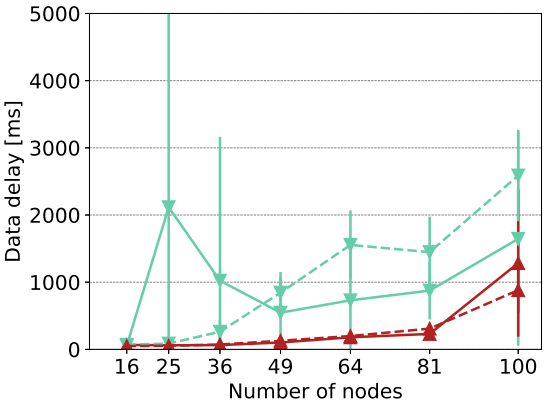
The Collect-based algorithm performs poorly in the presence of unidirectional links, regardless of link and topology type, presenting statistically different results from our solution.

The "controller to all" topology provides the

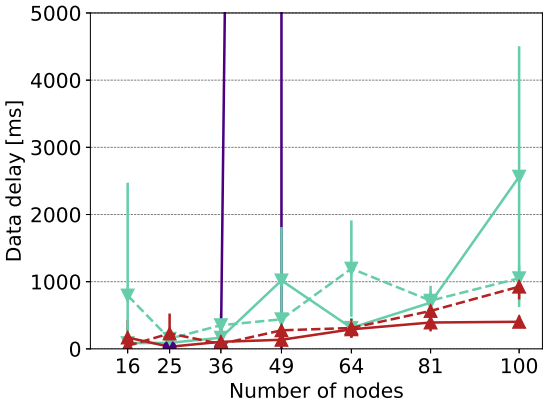
(a) Legend



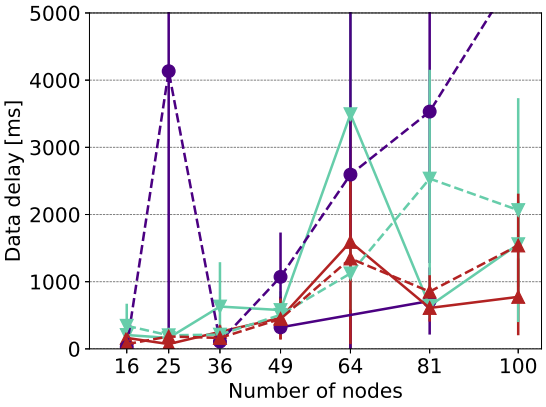
(b) Full



(c) Controller to all



(d) Random nodes with increased range



(e) Random unidirectional links

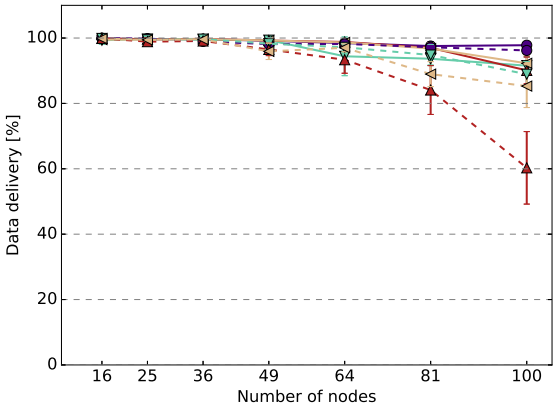
Figure 6: Pure CSMA: Data delay results.

largest delay improvement when unidirectional links are employed (Figure 6c). The improvement is at least 30% on large topologies, presenting statistically different results for most topology sizes (exceptions are 15-node and 100-node random topologies, and the 25-node grid). The unidirectional links from the controller enable a fast flow setup on all nodes. Conversely, the presence of unidirectional links, even if not used, increases link layer contention.

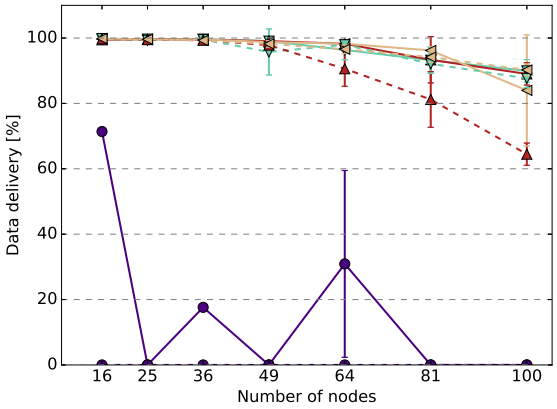
For the other link scenarios presented in Figure 6, the unidirectional links tend to decrease the average delay and standard deviation. Nonetheless, the delay metric is sensitive to the initial network transient, provoking standard deviation intervals overlap in many cases, which hinders the drawing of further conclusions.

The conclusion drawn from these results is that

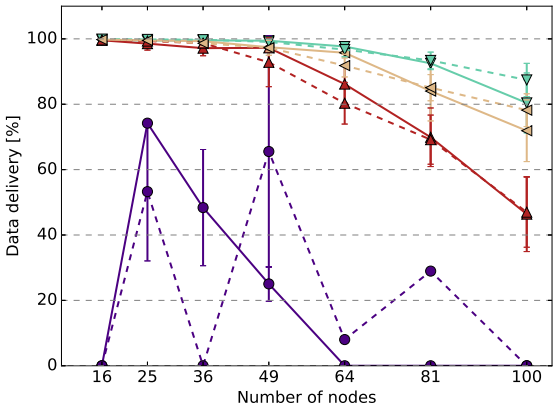
(a) Legend



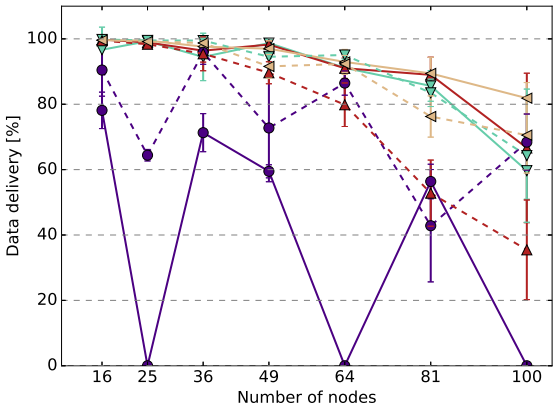
(b) Full



(c) Controller to all



(d) Random nodes with increased range



(e) Random unidirectional links

Figure 7: Duty cycled: Data delivery results.

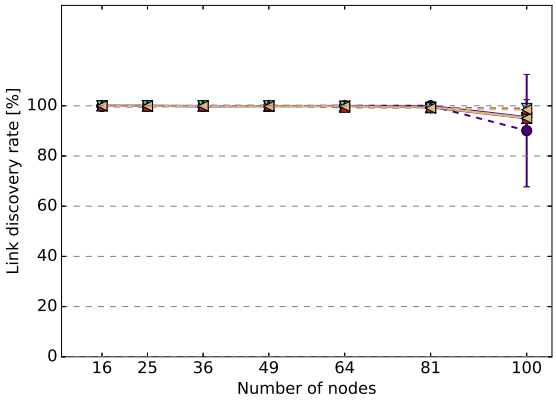
using unidirectional links positively effects packet delivery and tends to decrease the average data delay.

4.2. Duty cycling results

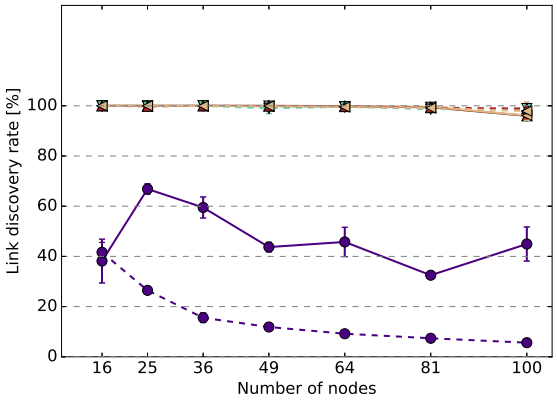
Data delivery results are displayed in Figure 7. The fully connected topology (Figure 7b) is useful for demonstrating the cost of supporting unidirectional links in comparison to solutions that do not support them. We observed that, in some of the simulation runs for large networks, a fraction of the links were initially detected as unidirectional, since the inbound and outbound components of a link are informed separately to the controller. As a consequence, if receivers are configured to send phase information directly to the sender, the controller sets and maintains control routes for communicating that phase information. This extra overhead degrades performance.

The addition of a link from the controller to each

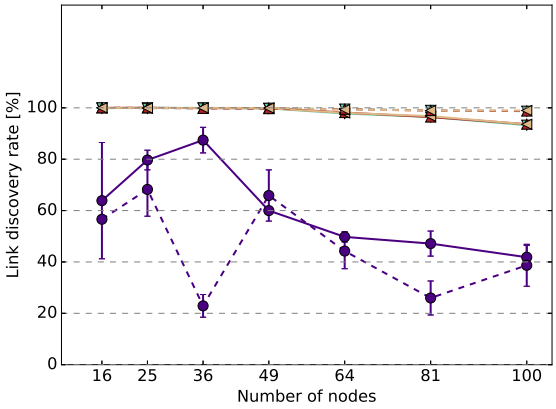
(a) Legend



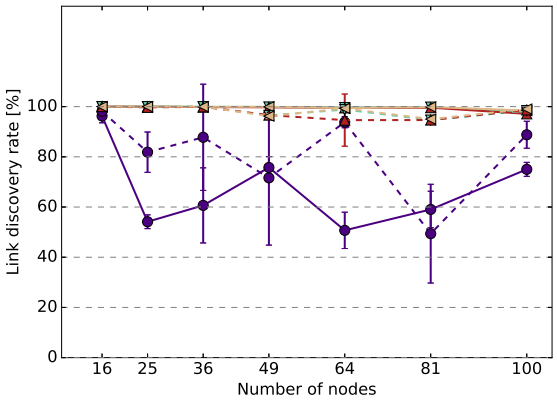
(b) Full



(c) Controller to all



(d) Random nodes with increased range



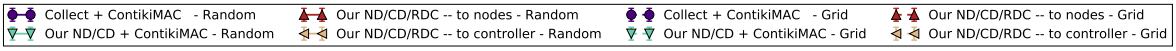
(e) Random unidirectional links

Figure 8: Duty cycled: Link discovery rate.

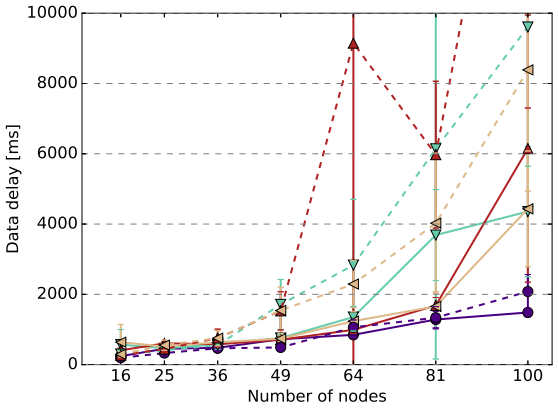
other node (Figure 7c) causes the Collect-based discovery protocol to fail, since it cannot correctly handle long-distance unidirectional links. There is no statistical difference between using our RDC and ContikiMAC, except in the case of large grid networks, wherein the send-to-nodes version of our RDC presented lower data yield.

If device heterogeneity causes some nodes to reach further than others, the RDC does not influence the delivery rate in small networks (Figure 7d). In large networks, the overhead of maintaining the phase information hinders rather than helps, as we observe a degradation in performance. For this type of topology, using the unidirectional link typically saves only one hop in the routing path, compared to the many hops that are saved in the "controller to all" topology.

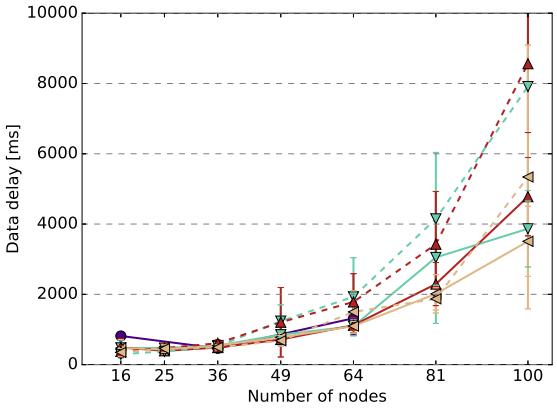
The "send to controller" version of our RDC yields small gains ($\approx 3\%$) in the random unidirectional links with random topology scenario (Fig-



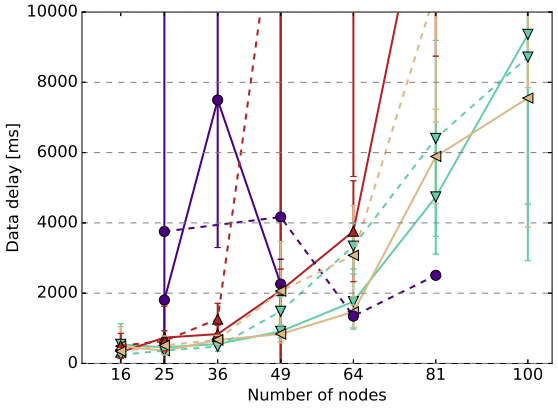
(a) Legend



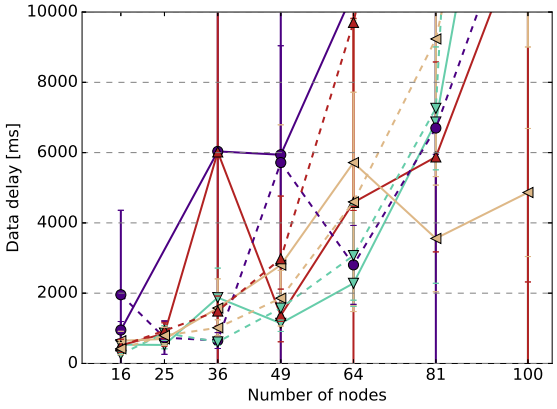
(b) Full



(c) Controller to all



(d) Random nodes with increased range



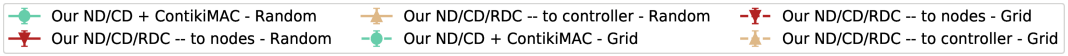
(e) Random unidirectional links

Figure 9: Duty cycled: Data delay results.

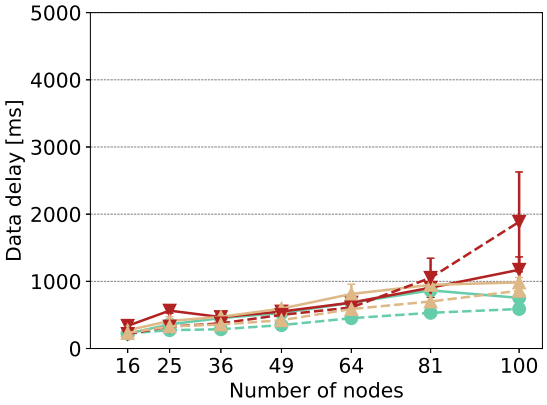
ure 7e), although the results are statistically different only for the 100-node network. The algorithm version that sends phase information directly to nodes tends to perform worse, presenting a statistically inferior data yield for the grid topology. The unidirectional links enable a larger number of possible routes in the network, alleviating link-layer congestion bottlenecks. This effect is more pronounced in the random network topology, since it contains more bottleneck nodes.

Figure 8 shows link discovery rate results. Our discovery protocols present consistent behavior in terms of detecting all links, regardless of the unidirectional link type. Eventually, links go undetected if the number of links exceeds the neighbor table capacity, which is caused by memory scarceness in network nodes.

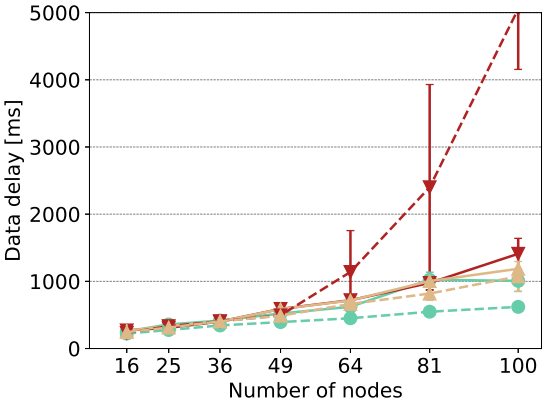
Regarding the Collect-based discovery protocol, the link discovery rate is greatly impacted by unidirectional links. This discovery protocol may mis-



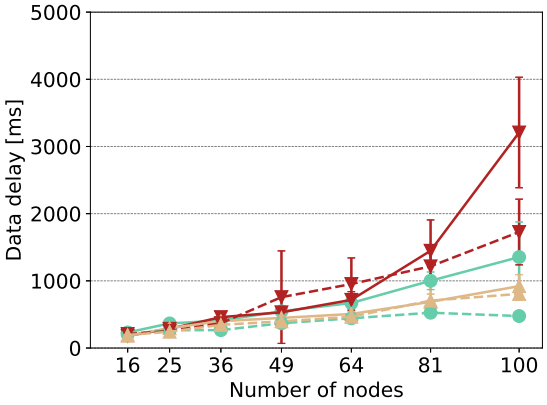
(a) Legend



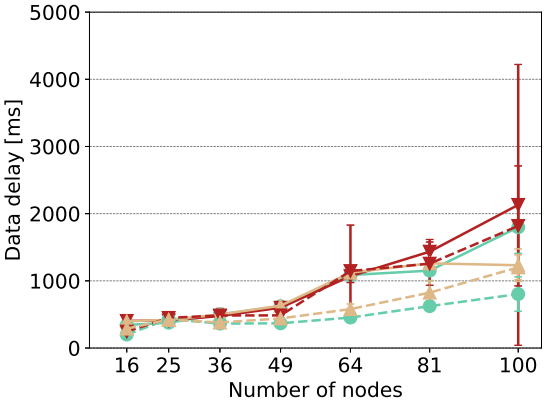
(b) Full



(c) Controller to all



(d) Random nodes with increased range



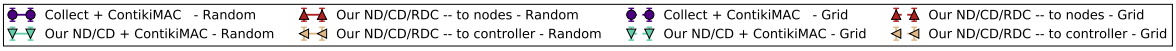
(e) Random unidirectional links

Figure 10: Duty cycled: Data delay after convergence results.

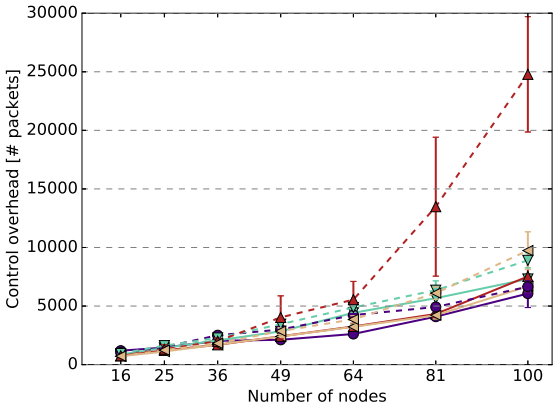
takenly assume that a unidirectional links is bidirectional, causing heavy losses in the network. Nodes farther from the controller might not be able to join the network, and therefore do not report their neighborhood status.

The most noticeable result for the data delay metric is observed with the "controller to all" topology (Figure 9c). Using long unidirectional links allows for faster initial route setup, decreasing the overall data delivery delay. The maximal improvement is 54.9%, for both grid and random topologies, although there is enough statistical evidence for only 49 and 81-node networks.

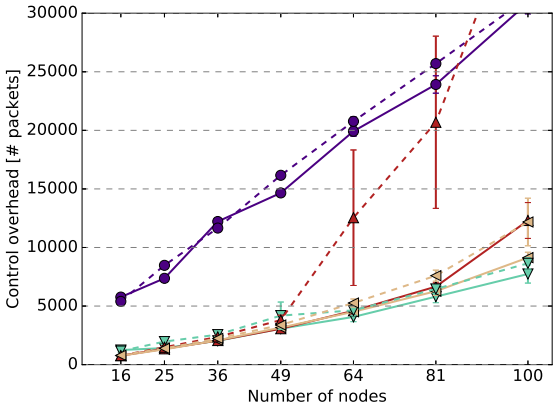
It is not possible to draw conclusions with respect to the other link scenarios. The delay metric is sensitive to the initial network setup, meaning that small differences in the packet ordering and collisions at this stage can have great influence on the average delay. Such circumstances account for the large standard deviation values, since some simula-



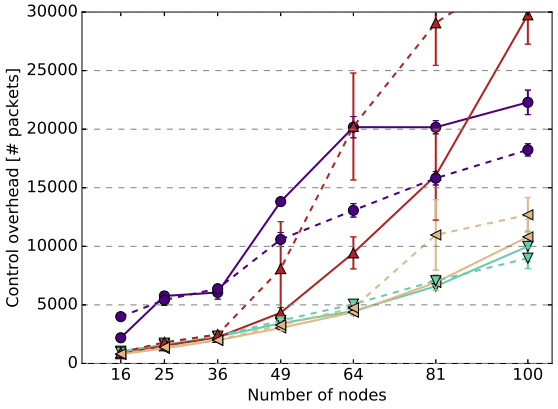
(a) Legend



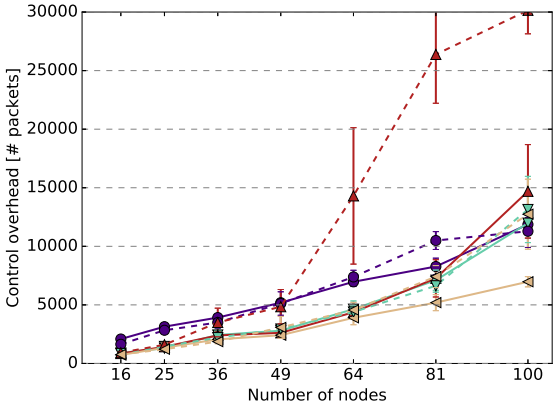
(b) Full



(c) Controller to all



(d) Random nodes with increased range



(e) Random unidirectional links

Figure 11: Duty cycled: Control overhead results.

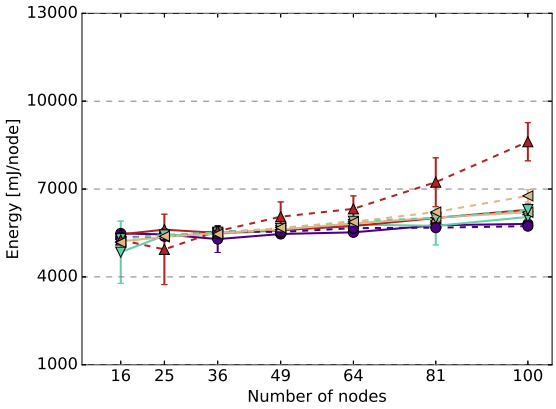
tions contain packets with a very high delay. This observation is confirmed by comparing the average to the delay median, the latter being significantly smaller in most cases. Nonetheless, there is a trend indicating that, while random topologies present a larger average delay in small networks, they are faster than grids in networks with more nodes.

To provide better understanding of the delay metric, we analyzed the delay of data packets transmitted after network convergence, shown in Figure 10. The convergence criterion is the time at which every node has successfully transmitted at least one data packet to the sink.

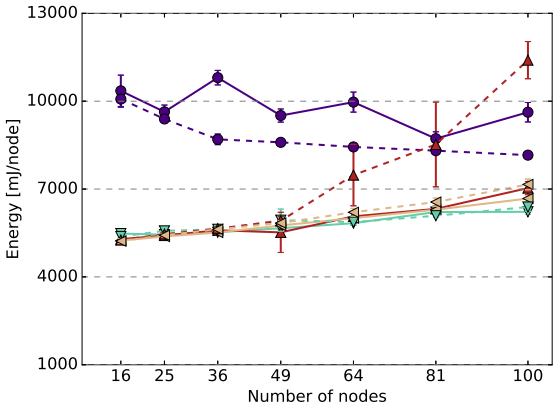
It is noticeable that the average delay after convergence is significantly smaller than the delay considering all packets, reassuring that the initial flow configuration has a great impact on this metric.

Additionally, the confidence intervals are narrower, indicating a consistent steady state behavior. Nonetheless, our approach that sends phase

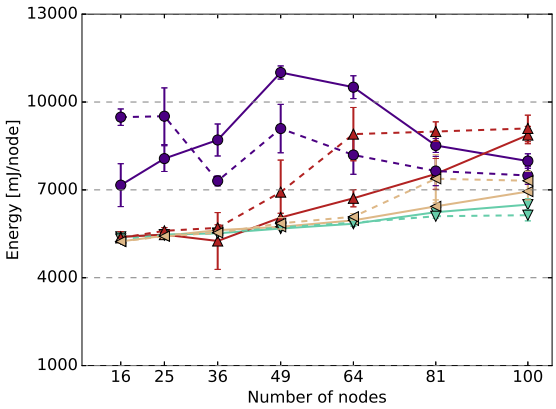
(a) Legend



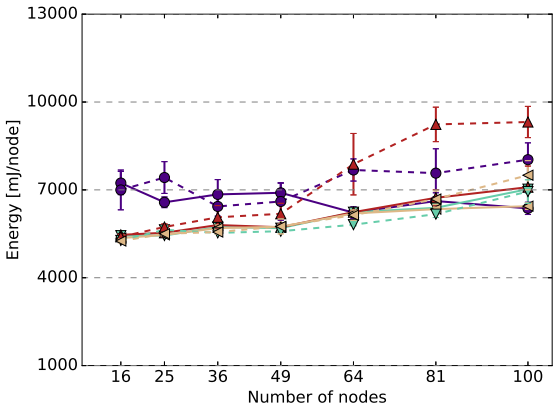
(b) Full



(c) Controller to all



(d) Random nodes with increased range



(e) Random unidirectional links

Figure 12: Duty cycled: Energy usage results.

information directly to nodes still presents large dispersion in large networks, as some packets are occasionally delayed after many seconds even after convergence.

The control overhead metric results are shown in Figure 11. On small and medium networks (up to 64 nodes), the combination of our ND and our RDC (send to controller variation) presents less overhead than the combination of our ND and ContikiMAC. The difference reaches 40% in some scenarios, mainly in the "controller to all" topology. This occurs due to the usage of unidirectional links, which decreases average route size. Shorter routes require fewer flow setup packets and are less prone to link layer errors (reducing retransmissions).

This advantage is less prominent in larger networks (81 and 100), due to the increased number of control messages used to keep the phase information of unidirectional link up-to-date.

Our RDC variation that sends phase information

directly to the nodes does not perform as good as the other variation. Nodes request control routes even for links temporarily detected as unidirectional at network start up. Even if these routes are not further used throughout the experiment, the controller keeps updating the routes according to variations of link quality, increasing the overall overhead. The larger the network, the more pronounced this effect.

The Collect-based approach is only more efficient in terms of control overhead in large fully connected topologies. It performs poorly in other link configurations, mostly due to low control packets delivery ratio and consequent retransmissions.

We expected that our RDC would reduce overall energy consumption, since using unidirectional links reduces the average path length. However, gains from shortening the average length path are ultimately negated by the extra overhead of maintaining control routes and sending phase information.

For most scenarios, there are no significant differences (less than $\pm 5\%$) in energy consumption between our ND combined with ContikiMAC and our ND combined with our RDC. Collect-based discovery combined with ContikiMAC also stays in that range when considering the topology with only bidirectional links. In large topologies, the large number of control packets raises the average energy consumption of both the Collect-based approach and our RDC variation that sends phase information directly to the nodes.

5. Related work

We review related work in three categories: 1) routing over unidirectional links; 2) discovery in Software-Defined WSN; and 3) radio duty cycling.

Routing over unidirectional links. Some researchers adopted the strategy of reducing the scope of the problem, yielding a solution that works only under specific conditions [8, 19]. Another strategy is to find an alternative path to the unidirectional link by flooding the network [29, 18].

Chen *et al.* proposed a probabilistic routing algorithm focused on many-to-one traffic [8]. The network is sliced into concentric stripes centered at the destination. Nodes transmit every packet multiple times, and the receivers forward with a certain probability. A node closer to the sink may overhear a packet and opportunistically transmit it to the sink, even if the link is unidirectional. The main drawbacks of this work are a restriction of applicability to many-to-one traffic and a wasting of resources due to multiple transmissions of the same packet [8].

Kim *et al.* assume a network with a single sink able to reach any node in the network in one hop, although most nodes cannot reach the sink in one hop. The objective is to provide efficient and reliable downward data transmission. However, the main assumption is difficult to generalize to a case where any link could be unidirectional [19].

Bidirectional Routing Abstraction (BRA) provides full support for unidirectional links by searching through the network for multihop reverse paths to the unidirectional links. It uses a Distributed Bellman–Ford algorithm, which floods the network

with distance vector information. To curb the control overhead, the maximum length of the reverse path is limited. The simulation results, based on the IEEE 802.11 standard, show that the AODV protocol performs better with BRA than when using the traditional blacklisting mechanism [29].

Unidirectional Link Counter (ULC) is actually a cross-layer protocol, as it mixes functionality from medium access and routing layers. The protocol is similar to AODV in the sense that Route Request and Route Response messages are used to discover routes, i.e., it is a flooding-based protocol. In addition, these messages are used to perform link discovery. If a link is unidirectional, the forwarding node relegates the forwarding task to its neighbors, expecting that at least one of them is able to detour the unidirectionality and find a reverse path. Both simulations and testbed results show performance enhancements when compared to AODV [18].

It is noteworthy that BRA and ULC are focused on MANETs and use flooding-based messages. In the context of WSN, it is desirable to avoid flooding in order to diminish the number of control packets throughout the network.

Our solution leverages the SDN paradigm to overcome the limitations of the previous work by providing a general-purpose and flooding-free protocol.

Discovery in Software-Defined WSN. Although several Software-Defined Wireless Sensor Networking frameworks exist in the literature, the neighbor discovery process is not detailed in most of them. The earlier proposals were adaptations of the Openflow protocol [24] to WSN, namely FlowSensor [22] and Sensor Openflow [21]. Neither of these two mentions the discovery process, although one could infer that they use a variation of the Link Layer Discovery Protocol (LLDP), since it is the OpenFlow default protocol.

TinySDN is a southbound protocol specification based on flow labeling, built on top of TinyOS ActiveMessage component [10]. Neighbor and controller discovery are relegated to the Collection Tree Protocol (CTP) [15], which in turn builds a tree rooted at the controller. Link quality is obtained from the TinyOS 4-bit link estimator, requiring bidirectional links. The authors do not detail the criteria for transmitting neighborhood information to the controller due to link quality variations.

IT-SDN is based on TinySDN, but with the goal of being OS-independent and to allow for replaceable discovery algorithms [23]. The default configuration is to use a CTP-like protocol for neighbor discovery, which is similar to TinySDN and presents the same drawbacks. A node sends topological information to the controller if the CTP link quality estimation differs more than 20% from the last reported value.

In a previous work, we introduced the naive neighbor and controller discovery algorithms for networks with unidirectional links [1]. We fixed the scalability issues, added a link quality estimation and neighbor unrechability detection [2]. We further detailed these mechanisms in Section 2.2.

SDN-Wise executes controller and neighbor discovery as a single operation by implementing its own topology discovery protocol [14]. The (possibly) multiple controllers start the construction of a tree by transmitting Topology Discovery packets. The tree is rebuilt periodically to obtain fresh

topology information. Nodes transmit topology information to the controller based on a fixed periodic interval. The authors analyzed the overhead of shortening this interval.

The four main shortcomings of the SDN-Wise discovery protocol are: 1) assuming links are bidirectional; 2) the unnecessary transmission of control packets due to the lack of adequate criteria to send topological information to the controller; 3) a fixed Topology Discovery transmission interval; and 4) the use of RSSI as link metric (hardware dependent). Hieu *et al.* introduced variable interval for topology discovery in SDN-Wise [16].

Theodorou and Mamatas proposed two discovery protocols, which they called "neighbor advertisement" and "neighbor request" [35]. The controller is responsible for starting either algorithm by transmitting a unicast message to its neighbors, which in turn broadcasts a discovery packet. In the "neighbor advertisement", each receiving node advertises the discovered link to the controller, while, in the "neighbor request", each receiving node answers back to the sender, which is responsible for reporting to the controller about the links. The controller subsequently transmits a unicast packet to the newly detected nodes, until all network nodes have been discovered. The researchers, however, do not specify how to perform the link quality assessment, how the controller discovery process occurs and how to deal with unidirectional links. In addition, their paper focuses only on the initial discovery phase and does not discuss when to re-collect the neighborhood information.

Our work improves on the existing neighbor discovery algorithms for SDWSN. We employ techniques to reduce the overall overhead associated with the task, while we properly support discovery of unidirectional links. We have also integrated link quality assessment and node departure detection, providing details on how to calculate the ETF and a criterion to send new topology update messages to the controller. Table 6 summarizes the main features of the discovery algorithms found in SDWSN frameworks.

Radio duty cycling. There exist two classes of RDC protocol: synchronous and asynchronous. Synchronous RDC requires a negotiation phase to allocate slots, elect cluster heads or keep tight clock synchronization, making it inherently complex. Furthermore, unidirectional links restrict such negotiations. This approach is recommended for high data rates or to meet Quality of Service requirements, but it has not been explored for unidirectional links. An example of synchronous RDC is the TSCH mode from IEEE 802.15.4e [17].

Asynchronous RDCs are further categorized into receiver initiated and sender initiated. Receiver initiated protocols are immediately discarded for supporting unidirectional links since they require a bidirectional interaction between sender and receiver. RI-MAC is the first work on receiver initiated RDCs [32], while other protocols built upon that same foundation [13, 33].

Basic sender initiated RDCs do not rely on a bidirectional links to work, but instead use long preambles which are not energy efficient. Enhancements were introduced [28, 7], with ContikiMAC being considered the state-of-the-art sender initiated RDC implementation, featuring packetized preambles, early acknowledgements and phase lock

Work	Approach	Unidir link	LQE	Criteria to controller	Neighbor departure
[10]	Collect	No	ETX	Undisclosed	Long timer
[23]	Collect	No	ETX	ETX variation	Long timer
[14]	Periodic	No	RSSI	Periodic	Periodic recollection
[35]	Controller	No	RSSI	Undisclosed	Undisclosed
[1]	Periodic	Yes	Hop	New nodes	Undisclosed
This work	Periodic + overhearing	Yes	ETF	LQE variation	Based on LQE

Table 6: SDWSN neighbor discovery comparison.

capability [11].

Nonetheless, to the best of our knowledge, there are no efficient RDCs suitable for networks with unidirectional links. The existing optimizations for asynchronous RDCs rely on acknowledgements, while synchronous RDCs require tight synchronization and a negotiation phase.

We have filled a dearth in the previously available literature by introducing an efficient asynchronous RDC that copes with unidirectional links. The solution leverages the SDN controller's global view of a network to distribute directionality and phase information.

6. Conclusion

The pervasiveness of unidirectional links in low-power wireless networks calls for a routing protocol that is able to cope with such links. In fact, beyond merely coping, the routing protocol could use the unidirectional links, potentially shortening route path lengths and saving resources. Therefore, we posed the following question: "What are the gains from exploring unidirectional links in low-power wireless networks?".

To answer this question, we provided an SDN-based centralized solution, including appropriate discovery protocols, and an asynchronous radio duty cycling protocol. This SDN-based solution deals with the two limitations found in previous work: a limited scope and the use of flooding. To the best of our knowledge, our RDC is the first asynchronous protocol tailored for unidirectional links.

We simulated a variety of network configurations, and have reached several conclusions. At minimum, unidirectional links must be detected and blacklisted, otherwise the packet delivery ratio drops significantly. Leveraging unidirectional links for routing bestows benefits if the unidirectional links are long, the greatest improvement being seen in the metric of packet delay. Minor data delivery gains were observed for short-ranged unidirectional links in medium-sized duty-cycled networks. In large duty-cycled networks, the additional control over-

head negates the effect of most achievable gains, to the extent where we did not observe any significant performance improvement. Conversely, non-duty-cycled networks always benefit from unidirectional links.

References

[1] R. C. A. Alves and C. B. Margi. Discovery protocols for SDN-based Wireless Sensor Networks with unidirectional links. In *XXXV Simpósio Brasileiro de Telecomunicações e Processamento de Sinais (SBrT)*, São Pedro, Brazil, 2017. Sociedade Brasileira de Telecomunicações.

[2] R. C. A. Alves, C. B. Margi, and F. A. Kuipers. No way back? An SDN protocol for directed IoT networks. In *15th Wireless On-demand Network systems and Services Conference*. IEEE, jan 2019.

[3] L. Atzori, A. Iera, and G. Morabito. The internet of things: A survey. *Computer Networks*, 54(15):2787 – 2805, 2010.

[4] N. Baccour, A. Koubâa, H. Youssef, and M. Alves. Reliable link quality estimation in low-power wireless networks and its impact on tree-routing. *Ad Hoc Networks*, 27(C):1–25, Apr. 2015.

[5] G. Barrenetxea, F. Ingelrest, G. Schaefer, and M. Vetterli. The hitchhiker's guide to successful wireless sensor network deployments. In *Proceedings of the 6th ACM Conference on Embedded Network Sensor Systems*, SenSys '08, pages 43–56, New York, NY, USA, 2008. ACM.

[6] S. Bindel, S. Chaumette, and B. Hilt. F-ETX: An Enhancement of ETX Metric for Wireless Mobile Networks. *Communication Technologies for Vehicles*, 9066:117–128, 2015.

[7] M. Buettner, G. V. Yee, E. Anderson, and R. Han. X-MAC: A short preamble MAC protocol for duty-cycled wireless sensor networks. In *Proceedings of the 4th International Conference on Embedded Networked Sensor Systems*, SenSys '06, pages 307–320, New York, NY, USA, 2006. ACM.

[8] X. Chen, Z. Dai, W. Li, and H. Shi. Performance guaranteed routing protocols for asymmetric sensor networks. *IEEE Transactions on Emerging Topics in Computing*, 1(1):111–120, June 2013.

[9] D. S. J. De Couto, D. Aguayo, J. Bicket, and R. Morris. A high-throughput path metric for multi-hop wireless routing. *Wireless Networks*, 11(4):419–434, July 2005.

[10] B. T. de Oliveira, C. B. Margi, and L. B. Gabriel. TinySDN: Enabling multiple controllers for software-defined wireless sensor networks. In *LATINCOM*, pages 1–6, Nov 2014.

[11] A. Dunkels. The ContikiMAC Radio Duty Cycling Protocol. Technical Report T2011:13, Swedish Institute of Computer Science, Dec. 2011.

[12] A. Dunkels, F. Osterlind, N. Tsiftes, and Z. He. Software-based on-line energy estimation for sensor nodes. In *Proceedings of the 4th workshop on Embedded networked sensors*, pages 28–32. ACM, 2007.

[13] P. Dutta, R. Musăloiu-e, I. Stoica, and A. Terzis. Wireless ACK collisions not considered harmful. In *HotNets-VII: The Seventh Workshop on Hot Topics in Networks*, 2008.

[14] L. Galluccio, S. Milardo, G. Morabito, and S. Palazzo. SDN-WISE : Design , prototyping and experimentation of a stateful SDN solution for WIreless SEnsor networks. *2015 IEEE Conference on Computer Communications (INFOCOM)*, pages 513–521, 2015.

[15] O. Gnawali, R. Fonseca, K. Jamieson, D. Moss, and P. Levis. Collection tree protocol. In *Proceedings of the 7th ACM Conference on Embedded Networked Sensor Systems*, SenSys '09, New York, NY, USA, 2009. ACM.

[16] N. Q. Hieu, N. Huu Thanh, T. T. Huong, N. Quynh Thu, and H. V. Quang. Integrating trickle timing in software defined wsns for energy efficiency. In *2018 IEEE Seventh International Conference on Communications and Electronics (ICCE)*, pages 75–80, July 2018.

[17] IEEE. 802.15.4e-2012 - IEEE Standard for Local and metropolitan area networks–Part 15.4: Low-Rate Wireless Personal Area Networks (LR-WPANs) Amendment 1: MAC sublayer, 2012.

[18] R. Karnapke and J. Nolte. Unidirectional link counter - a routing protocol for wireless sensor networks with many unidirectional links. In *Ad Hoc Networking Workshop (MED-HOC-NET), 2015 14th Annual Mediter-*

ranean, pages 1–7, June 2015.

[19] H.-S. Kim, M.-S. Lee, Y.-J. Choi, J. Ko, and S. Bahk. Reliable and energy-efficient downward packet delivery in asymmetric transmission power-based networks. *ACM Transaction on Sensor Networks*, 12(4):34:1–34:25, Sept. 2016.

[20] K.-H. Kim and K. G. Shin. On accurate measurement of link quality in multi-hop wireless mesh networks. In *Proceedings of the 12th Annual International Conference on Mobile Computing and Networking*, MobiCom '06, pages 38–49, New York, NY, USA, 2006. ACM.

[21] T. Luo, H.-P. Tan, and T. Q. S. Quek. Sensor openflow: Enabling software-defined wireless sensor networks. *IEEE Communications Letters*, 16(11):1896–1899, 2012.

[22] A. Mahmud and R. Rahmani. Exploitation of openflow in wireless sensor networks. In *Computer Science and Network Technology (ICCSNT)*, volume 1, pages 594–600, 2011.

[23] C. B. Margi, R. C. A. Alves, G. A. N. Segura, and D. A. G. Oliveira. Software-defined wireless sensor networks approach: Southbound protocol and its performance evaluation. *Open Journal of Internet Of Things (OJIOT)*, 4(1):99–108, 2018. Special Issue: Proceedings of the International Workshop on Very Large Internet of Things (VLIoT 2018) in conjunction with the VLDB 2018 Conference in Rio de Janeiro, Brazil.

[24] N. McKeown, T. Anderson, H. Balakrishnan, G. Parulkar, L. Peterson, J. Rexford, S. Shenker, and J. Turner. OpenFlow: Enabling innovation in campus networks. *SIGCOMM Comput. Commun. Rev.*, 38(2):69–74, Mar. 2008.

[25] B. Milic and M. Malek. NPART - Node Placement Algorithm for Realistic Topologies in Wireless Multihop Network Simulation. In *Proceedings of the 2Nd International Conference on Simulation Tools and Techniques*, Simutools '09, pages 9:1–9:10, 2009.

[26] F. Osterlind, A. Dunkels, J. Eriksson, N. Finne, and T. Voigt. Cross-level sensor network simulation with COOJA. In *Proceedings. 31st IEEE Conference on Local Computer Networks*, Nov 2006.

[27] C. Perkins, E. Belding-Royer, and S. Das. Ad hoc On-Demand Distance Vector (AODV) Routing. RFC 3561, IETF, July 2003.

[28] J. Polastre, J. Hill, and D. Culler. Versatile low power media access for wireless sensor networks. In *Proceedings of the 2nd International Conference on Embedded Networked Sensor Systems*, SenSys '04, pages 95–107, New York, NY, USA, 2004. ACM.

[29] V. Ramasubramanian and D. Mosse. BRA: A Bidirectional Routing Abstraction for asymmetric mobile ad hoc networks. *IEEE/ACM Transactions on Networking*, 16(1):116–129, Feb 2008.

[30] L. Sang, A. Arora, and H. Zhang. On link asymmetry and one-way estimation in wireless sensor networks. *ACM Transactions on Sensor Networks*, 6(2):12:1–12:25, Mar. 2010.

[31] T. Shibata, R. de Azevedo, B. C. Albertini, and C. B. Margi. Energy consumption and execution time characterization for the SensorTag IoT platform. In *XXXIV Simpósio Brasileiro de Telecomunicações e Processamento de Sinais (SBrT 2016)*, pages 55–59, Santarém, Brazil, Aug. 2016.

[32] Y. Sun, O. Gurewitz, and D. B. Johnson. RI-MAC: A receiver-initiated asynchronous duty cycle MAC protocol for dynamic traffic loads in wireless sensor networks. In *Proceedings of the 6th ACM Conference on Embedded Network Sensor Systems*, SenSys '08, pages 1–14, New York, NY, USA, 2008. ACM.

[33] L. Tang, Y. Sun, O. Gurewitz, and D. B. Johnson. PW-MAC: An energy-efficient predictive-wakeup MAC protocol for wireless sensor networks. In *2011 Proceedings IEEE INFOCOM*, pages 1305–1313, April 2011.

[34] Texas Instruments. *CC2420 2.4 GHz IEEE 802.15.4 / ZigBee-ready RF Transceiver*, 2004. (Rev. C).

[35] T. Theodorou and L. Mamatas. Software defined topology control strategies for the internet of things. In *2017 IEEE Conference on Network Function Virtualization and Software Defined Networks*, (NFV-SDN), Nov 2017.

[36] A. Woo and D. Culler. Evaluation of efficient link reliability estimators for low-power wireless networks. Technical Report UCB/CSD-03-1270, EECS Department, University of California, Berkeley, 2003.

[37] A. Woo, T. Tong, and D. Culler. Taming the underlying challenges of reliable multihop routing in sensor

networks. In *Proceedings of the 1st International Conference on Embedded Networked Sensor Systems*, SenSys '03, pages 14–27, New York, NY, USA, 2003. ACM.

[38] W. Ye, J. Heidemann, and D. Estrin. An energy-efficient MAC protocol for wireless sensor networks. In *Proceedings.Twenty-First Annual Joint Conference of the IEEE Computer and Communications Societies*, volume 3, pages 1567–1576 vol.3, June 2002.