TagAlong: Efficient Integration of Battery-free Sensor Tags in Standard Wireless Networks

Carlos Pérez-Penichet
Uppsala University
Sweden
carlos.penichet@it.uu.se

Dilushi Piumwardane
Uppsala University
Sweden
dilushi.piumwardane@it.uu.se

Christian Rohner
Uppsala University
Sweden
christian.rohner@it.uu.se

Thiemo Voigt
Uppsala University and RISE SICS
Sweden
thiemo@sics.se

ABSTRACT

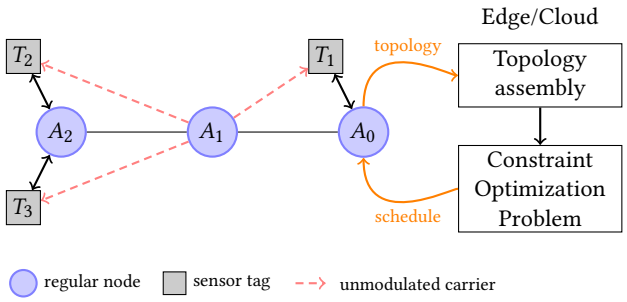
New battery-free sensor tags that interoperate with unmodified standard IoT devices can extend a sensor network's capabilities in a scalable and cost-effective manner. The tags achieve battery-free operation through backscatter-related techniques, while the standard IoT devices can provide the necessary unmodulated carrier, avoiding additional dedicated infrastructure. However, this approach presents multiple challenges: It requires coordination between nodes transmitting, receiving and generating carrier, adds extra latency and energy consumption to already constrained devices, and increases interference and contention in shared spectrum. We present TagAlong, a medium access mechanism for interoperable sensor tags that, besides coordinating, optimizes the use of carrier generators, minimizing the disruption caused to the operation of the regular nodes. We accomplish this by parallelizing communications with battery-free tags when possible, sharing carriers for multiple tags concurrently and synchronizing communications with tags that share carrier generators. We demonstrate the feasibility of TagAlong in a testbed deployment. In our evaluation we find that it can reduce the duration of the tags' schedule by 60% while improving the energy and spectrum usage by 30% when compared to sequential interrogation with no difference in reliability.

CCS CONCEPTS

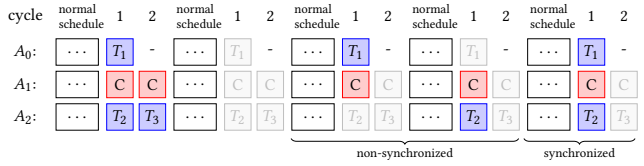
• **Networks** → **Link-layer protocols**; **Physical topologies**; • **Hardware** → *Wireless integrated network sensors.*

KEYWORDS

Battery free sensors, Backscatter, Internet of Things, Scheduling



(a) Example of network topology and system model showing when node A_1 emits and unmodulated carrier to interrogate tags.



(b) TagAlong's optimized schedule. Interrogations are synchronized to maximize reuse of unmodulated carrier slots (C).

Figure 1: *TagAlong ensures the availability of the unmodulated carrier that sensor tags need to communicate.* We optimize tag interrogations sharing unmodulated carriers for multiple tags and interrogating them in parallel. This reduces latency, power consumption and spectrum usage compared to sequential interrogation.

1 INTRODUCTION

Recent progress in battery-free communications enable new devices that can both transmit and receive standard IoT physical layer protocols with a power consumption well under 1 mW, while assisted by an external unmodulated carrier [10, 26, 39, 40]. These new devices interoperate with unmodified IoT nodes with standard hardware and communication protocols. Meanwhile, their drastically reduced power consumption allows them to leverage a broad range of energy harvesting techniques [4] to operate indefinitely without batteries. As a consequence, they can be an attractive complement to regular sensor nodes in novel applications [15, 17] including localization [23, 29, 33, 35], sensors embedded in the infrastructure [15, 44], medical implants or wearable devices [21], where having batteries in all nodes may be impractical.

Scenario. *Sensor tags* are battery-free stickers, similar to RFID tags, containing their own sensors. These devices can extend the sensing capabilities of existing IoT deployments as easily as when installing new wireless peripherals to our computers. One would place them next to the existing IoT nodes which then coordinate to interrogate the tags for their measurements [39, 40]. This way we avoid the need for dedicated readers and carrier generators; we reduce downtime, maintenance and deployment costs associated with deploying new sensors and batteries; and we avoid costly hardware modifications to the existing infrastructure. Sensor tags can also enable

the installation of sensors in hard to reach places, where batteries or wires may not be viable, while keeping battery-powered nodes in nearby accessible locations for easy maintenance. For example, tags could be embedded in the infrastructure, covered in concrete or implanted in humans or animals, where measurements must be taken, while keeping battery-powered devices accessible outside to collect and process sensor readings.

Challenges. Battery-free sensor tags require an external unmodulated carrier to communicate [39, 40]. In scenarios like the above, it is desirable that the regular IoT nodes provide the carrier to avoid adding dedicated devices. This approach puts an additional burden on already constrained sensor nodes, as they invest energy in carrier generation and reserve time for tag communications, which degrades latency network-wide. Therefore, sensor nodes must use the carrier efficiently to avoid unnecessarily occupying the medium and draining their batteries. Providing efficient on-demand carrier support requires careful orchestration: During tag interrogations we must coordinate a carrier generator, the interrogator and the tag while avoiding network-wide collisions, as well as unnecessary access and communication latency. This is challenging in three specific ways: First, we must compute an efficient schedule that allows interrogating every tag with minimum resource investment from the sensor nodes. Second, to ensure that carriers are only used when there is demand, interrogating nodes must transmit a carrier activation request. Hence, shared carrier generators must be able to respond to concurrent carrier activation requests which may collide and be lost. Third, we must ensure that tags sharing a carrier generator are interrogated at the same time to avoid unnecessarily activating the same carrier multiple times to serve different tags.

Contribution. We present TagAlong, a medium access mechanism for heterogeneous networks of standard nodes interoperating with battery-free sensor tags, as illustrated in Figure 1(a). TagAlong automatically coordinates carrier generators, transmitters and receivers in communications with sensor tags. Beyond coordination, we optimize both the tag interrogation schedule and the interrogation protocol, as illustrated in Figure 1(b), to reduce latency, power consumption and spectrum usage.

We make the following key contributions:

- Optimize carrier scheduling to interrogate multiple tags concurrently, sharing carrier generators whenever possible. This avoids disruption to regular nodes in terms of communication latency, power consumption and spectral usage.
- Introduce a carrier request mechanism that enables concurrent requests without collisions when multiple nodes share a carrier generator.
- Propose a mechanism that synchronizes interrogations of tags that share carrier generators to make sure we do not activate a carrier unnecessarily, thereby saving energy and spectrum as shown in Figure 1(b).
- We demonstrate and evaluate our system with an implementation in an IEEE 802.15.4 testbed, showing no decrease in reliability with our scheme. A systematic analysis with testbed topologies evaluates its scalability and system performance at a larger scale.

Approach. We adopt a time-slotted protocol to pre-assign the function of every device during every time slot and coordinate transmitters, receivers and carrier generators. Communications among regular nodes are scheduled independently of TagAlong, using existing slot assignment mechanisms [8, 28]. Additional slots are then appended to the original schedule to perform tag interrogations (c.f., Figure 1(b)). The key idea of TagAlong is to reduce the duration of the tag interrogation schedule by leveraging on regular nodes that can operate as carrier generators for multiple tags at the same time, and by parallelizing interrogations when possible. Shortening the schedule in this way reduces latency, both in communications among regular nodes and to sensor tags, and reduces the energy consumption of the system. Sharing a carrier generator for multiple simultaneous tag interrogations is possible because the communication range of battery-free tags is short when compared to standard IoT nodes, which enables spatial reuse (Figure 2).

We approach tag scheduling as a centralized Constraint Optimization Problem (COP). The COP takes the topology of the network as input and computes a schedule that is optimal in that it requires the *least time*, and the *fewest carrier generators* to interrogate every tag in the network without collisions, as exemplified in Figure 1. Tag interrogation slots are used only on request to avoid unnecessary power consumption and spectrum use. Tags are too constrained to follow the schedule, instead they are interrogated on demand. Finally, to maximize carrier sharing, and therefore energy savings, we synchronize sensor tag interrogations that use the same carrier generator.

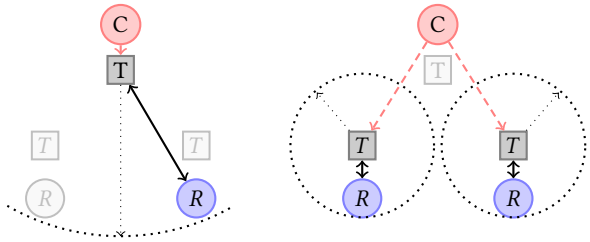
Results. Our testbed experiments show that TagAlong provides the unmodulated carrier whenever it is needed. The system operates efficiently in terms of energy and spectrum usage, and without unnecessary overhead. Our evaluation shows that, for the largest number of tags evaluated, we can obtain a 60% decrease in the duration of the tags' schedule, and a 30% reduction in the number of necessary carriers relative to sequential interrogation, a common alternative approach found in the literature [21, 26]. These results lead to significant improvement in latency and energy savings with no discernible reliability penalty.

Outline. The rest of the paper is organized as follows: In Section 2 we present the necessary background to understand the rest of the paper. In Section 3 we discuss the design aspects of TagAlong. In Section 4 we present our implementation and in Section 5 we evaluate its performance. Section 6 compares TagAlong to related approaches. In Section 7 we discuss further improvements to our system and Section 8 exposes our conclusions.

2 BACKGROUND

Ultra-low power battery-free transceivers such as sensor tags use an external unmodulated carrier to transmit and to receive. Offloading the carrier to an external device is the key enabler for ultra-low power consumption. To transmit, the device employs backscatter communications and for reception it uses a receiver with an external Local Oscillator (LO). We now present the operating principles of each of the two techniques.

Backscatter transmitters. With a power consumption up to three orders of magnitude lower than traditional radios, backscatter transmitters selectively reflect an external Radio Frequency (RF) signal



(a) Tag close to carrier generator: Large collision domain.

(b) Tag close to receiving node: Small collision domain.

Figure 2: *The interference range of a tag (T) depends on the distance to the carrier generator (C).* We leverage this property to limit the tags' collision domain to a small region next to the receiver (R) which increases spatial reuse.

to convey information [26, 32, 47]. The backscatter transmitter controls the way incident RF is reflected by changing the load attached to its antenna to create a specific impedance mismatch. A receiver can decode the information by observing the changes in the reflected signal. Because the amplitude, phase and frequency of the reflected signal can be controlled this way, it is possible to backscatter almost any standard physical layer protocol. Multiple examples with WiFi [26, 47], Bluetooth [9], IEEE 802.15.4 [26, 39] and LoRa [16, 37, 43] have become available in the open literature in recent times.

Frequency-shifted backscatter. A crucial aspect when using backscatter with commodity devices is that the backscattered signal can be at a different frequency than the unmodulated carrier. The backscatter transmitter prevents the carrier from interfering at the receiver by placing its signal in a different channel [21, 47]. TagAlong leverages this technique to avoid interference from carrier-generating nodes. Because of this frequency separation, communications with tags occupy two channels [21, 39, 40].

Battery-free receiver. An external carrier can help a receiver operate with a power consumption well under 1 mW, in a way analogous to backscatter [10, 40]. A receiver of this kind sidesteps power-hungry blocks present in traditional radio receivers such as LOs and Analog-to-Digital Converters (ADCs) by replacing them with passive circuits whenever possible. Similar to backscatter transmitters, the receiver offloads the LO to an external device that broadcasts an unmodulated carrier with a frequency that is lower than that of the received signal. The receiver then employs a passive diode mixer to downconvert the RF signal to a low Intermediate Frequency (IF), where it can be treated more energy efficiently than the high-frequency signal.

Communication range and interference. Communication range in battery-free devices depends on the strength of the external carrier. This is a behavior observed both in backscatter transmitters [3, 26] and in receivers with an external LO [40]. The closer the device is to the carrier generator the longer the communication range and, as a consequence, the larger its collision domain becomes. Figure 2 illustrates how using a far away carrier leads to a short communication range for tags. TagAlong exploits this short range that limits the tags' collision domain as a form of spatial reuse to share carrier generators among multiple tags and to interrogate multiple tags in parallel.

Tags cannot operate properly while provided with multiple unmodulated carriers as the random phase and frequency offsets among the carriers would cause problems for both transmission and reception. We avoid this situation altogether in our system with the COP.

3 DESIGN OF TAGALONG

TagAlong's goal is to provide the unmodulated carrier for all tags in an efficient manner, disrupting the operation of the regular nodes as little as possible, as illustrated in Figure 1. At a high level, TagAlong performs the following steps: First, we collect the link-layer topology of the network of regular nodes in the cloud or edge server (Section 3.3). Second, we add the tags to the topology. Tag discovery is outside of the scope of this work. For now we follow a static mapping of tags to their corresponding regular node host. We then compute a schedule that is optimal in the sense that it requires the *least time*, and the *fewest carrier generators* to schedule every tag in the network once per slotframe (Section 3.4). This involves ensuring that there are no collisions and deciding on a suitable carrier generator node that each tag will use to receive interrogations and to transmit the reply. To minimize the length of the tags' schedule, energy consumption and spectrum usage, we choose a carrier that is suitable for multiple tags whenever possible, and interrogate as many tags as possible in parallel. Examples of topology and resulting schedules are shown in Figure 4. Third, we disseminate the schedule to all nodes in the network. During runtime, we can continue to update the topology information and recompute the schedule to account for link variability.

During operation, interrogating hosts must request the designated carrier generating node to transmit an unmodulated carrier, but concurrent requests to a shared carrier generator would collide and be lost. To solve this issue, hosts signal their need for the unmodulated carrier transmitting a short carrier themselves (Section 3.5). The carrier generating node can detect this signalling using its Clear Channel Assessment (CCA) mechanism. To take maximum advantage of carrier reuse, TagAlong synchronizes tag interrogations so that the ones with shared carrier generators happen at the same time (Section 3.6). Optimizing the schedule can be computationally intensive. We introduce a heuristic that reduces computation time by reducing the size of the scheduling problem to solve (Section 3.7).

We now discuss each of these aspects in more detail.

3.1 System Model

TagAlong operates in a heterogeneous network, like the one in Figure 1(a), comprising N_a regular nodes and N_t tags. Regular nodes are standard IoT devices with radio transceivers that support commodity physical layer protocols such as Bluetooth or IEEE 802.15.4 and have a means to generate an unmodulated carrier at a chosen frequency, for instance using their radio test mode [39] or other means [21]. The network of regular nodes runs on a TDMA Medium Access Control (MAC) protocol. We adopt a centralized approach assuming that at least one of the regular nodes in the network has

a connection to an Edge/Cloud server where we compute the optimal schedule for tag interrogations. This is because computing the schedule requires considerable computational resources. Regular nodes perform their own sensing and can provide services like routing, computation, and edge/cloud access.

Sensor tags are extremely constrained devices with a power consumption well under 1 mW that may operate on energy harvested from their environment. Harvesting modalities may include –but are not limited to– RF. Sensor tags are equipped with ultra-low power transceivers that are compatible with the same physical layer protocol used by the regular nodes, but need an external unmodulated carrier to transmit and receive, as described in Section 2. Tags employ frequency-shifted backscatter in their transmissions so carriers do not interfere with the backscattered signal at the receiver. Every sensor tag is placed next to a regular node within a radius of 15 cm to 25 cm approximately. The corresponding regular node for a tag is said to be its *host* and a regular node may host multiple sensor tags. The proximity of the tag to the host allows the carrier generating node to be located several meters away due to the communication range behavior discussed in Section 2.

The tags' low power nature allows them to operate in receive mode almost continuously from their harvested energy source, with a virtually-infinite lifetime [22]. As a consequence of their low power design, they can only perform very simple operations such as replying to an interrogation after a short time interval, i.e., similar to an RFID tag or to a computational RFID tag such as Wisp [1] and Moo [2]. We assume that tags are not able to maintain synchrony with the network of regular nodes for a long time, as doing so would require power hungry time-keeping circuitry.

3.2 Overall Design

During the tags' schedule, all regular nodes remain in an energy-preserving sleep mode with their radios off, only to become active when they are needed to generate a carrier or interrogate a tag. To this end, TagAlong divides time into discrete timeslots that are grouped into slotframes, which repeat periodically. The regular nodes' schedule is independently assigned by any of the existing mechanisms [8, 28] without knowledge of the tags. TagAlong then appends dedicated sensor tag interrogation cycles. During these cycles a host can transmit an interrogation and receive the reply while a suitable carrier generator is active nearby.

Interrogation cycles must be compatible with regular communications because the tags' schedule is appended to the regular one. We spread tag interrogation cycles across two consecutive timeslots as illustrated in Figure 3. The tags' downlink communications happen in one slot and the uplink in the slot immediately after. This way it is possible to support full-length frames in both directions without affecting the time slot duration for the regular nodes.

During operation, carrier timeslots are only needed if the application running at the hosts requires them to interrogate a tag. In sensing and monitoring applications, which are the main focus of TagAlong, most timeslots would remain unused until a tag must be interrogated. It would be wasteful to enable carrier generators on every assigned slot, as most of the times that would be unnecessary. As a consequence, interrogating hosts must request the activation

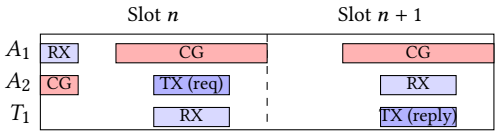


Figure 3: *Spreading interrogation cycles over two timeslots prevents disruption to the regular schedule.* The interrogating regular node (A_2) transmits a short carrier to request the carrier generator (A_1) to engage. With the carrier enabled, A_2 sends a request to the tag (T_1). The tag replies during the next time slot.

of the carrier generator to save energy and spectrum. We can configure TagAlong to unconditionally provide the unmodulated carrier at the appropriate times, thus avoiding the need to transmit and listen for carrier requests, in cases where we only intend to support periodic sensing.

Figure 3 shows the procedure that nodes follow to interrogate a sensor tag. A node that wants to interrogate one of its tags broadcasts a request at the beginning of the slot allocated to that tag. Nodes listen to the channel during timeslots when they are scheduled as carrier generators. When a request arrives, the carrier generator (A_1) enables the carrier long enough to allow for the transmission of one full-length frame. The interrogating node (A_2) then transmits its request addressed to the desired tag (T_1) that will be able to receive it since an unmodulated carrier is provided. After a delay, the interrogating node will be listening to receive the response from the tag. If the reply does not arrive, because either the carrier request or one of the messages was lost, or because the tag did not have enough energy, the interrogating node will have to try again later.

3.3 Network Topology Discovery

In order to ensure there are no collisions among transmissions (carriers or data) and that transmissions reach the intended destinations with sufficient signal strength, TagAlong needs to know the network topology. We represent the network topology as a heterogeneous undirected graph $G = (V, E)$ like the one in Figure 1(a). The vertex set $V = A \cup T$ comprises the set of regular nodes A and the tags T. The edge set E models radio links among devices. The weights of the edges $W_{i,j}$ represent the link quality metric observed between nodes i and j.

To build the topology graph, TagAlong follows three steps: First, it discovers the network topology of the regular IoT nodes and collects it at the server. To this end, we can employ one of many topology discovery mechanisms available in the literature [5, 24, 36, 42]. Second, TagAlong associates every tag to its host following the tag to host assignment mapping $H_t : t \in T \mapsto n \in A$ that is known a priori, either defined statically or from a tag discovery mechanism that we leave to future work. We add a link between every battery-free tag and its corresponding host. Third, we add links between battery-free tags and every one of the neighbors of its hosts setting the weights of the edges connected to every tag equal to the weights between the host and each of its neighbors. More formally:

$$W_{t \in T, n \in A} = W_{H_t, n} \tag{1}$$

This is necessary because battery-free tags generally lack link quality estimation capabilities. Measurements available in previous works [39, 40] show that tag communication success correlates to carrier strength at the host; therefore it is reasonable to use the signal strength at the host as a proxy for the strength at the tag in this way.

With the completed topology graph, we can compute the schedule with our COP. During runtime, we can continue to update the topology information and recompute the schedule to account for link variability.

3.4 Optimized Carrier Generator Scheduling

The straightforward way of scheduling sensor tag interrogations without collisions is to sequentially assign a dedicated slot to each tag in the network. This approach, however, requires unnecessarily long slotframes as the number of tags grows, leading to excessive communication latency and reduced network capacity for the whole network. Instead, we use the network topology to optimize the tag interrogation schedule.

Intuitively we see that we can leverage their short communication range to concurrently interrogate tags located far from each other without collisions, as in Figure 4(c). This saves interrogation cycles in the schedule, and therefore reduces latency. In fact, we can exploit the dependency of the tags' interference range with carrier strength to interrogate multiple tags using the same carrier generator, as illustrated in Figure 2(b). As shown in Figures 4(a) and 4(b), this reduces the number of necessary carrier generator slots, which decreases energy consumption and spectrum usage, besides also reducing latency.

Next we show how we can leverage these ideas to significantly reduce interrogation latency, the energy invested in generating carriers and spectrum usage. The objective in our COP is to find a slot allocation so that all battery-free tags in the network can be interrogated once per slotframe with the lowest possible number of carrier generator slots and in the shortest possible time. The solution to our COP (decision variables) is a schedule $S_{d,c} \in \{O, I, C\}$ and a destination address matrix $D_{d,c} \in V$. The schedule indicates the function assignment for every device d (tag or regular node) in the topology graph vertex set V ($d \in V$) in every interrogation cycle $c \in [1, N_t]$. An interrogation cycle consists of an interrogation request, reply pair across two consecutive timeslots as seen in Figure 3. The possible functions assigned to the devices are: Interrogate or be interrogated (I), generate an unmodulated carrier (C) and remain off (O). Note that the interrogation cycles are in the closed interval $[1, N_t]$ because, in the worst case (the sequential schedule), we need one interrogation cycle per tag.

The destination address matrix indicates the destination address for interrogation cycles performed by device d during cycle c. In our COP, we aim to minimize both the duration of the schedule and the number of carrier generation cycles. We employ linear scalarization to express both partial objectives as a single one: $Minimize\,(J_1 + (N_t + 1)\,J_2)$. Partial objective J_1 is the minimal number of cycles needed to complete the schedule, while J_2 counts the cycles dedicated to carrier generation.

We express J_1 in Equation 2. J_1 equals the cardinality (number of elements) of the set of cycles in the schedule where at least one of the devices is not off:

$$J_1 = |\{c \in [1, N_t] \colon \exists_{d \in V} S_{d,c} \neq O\}| \tag{2}$$

Equation 3 expresses partial objective J_2 as the cardinality of the set of cycles, over all the regular nodes, where the schedule is set to generate a carrier:

$$J_2 = |\{S_{d \in A, c \in [1, N_t]} \colon S_{d,c} = C\}| \tag{3}$$

Admissible solutions to our COP must concurrently satisfy all of the following constraints during every tag interrogation cycle:

- **C1** There must be one and only one neighboring regular node in carrier generator (C) mode. The link metric from the carrier to the tag must overcome a threshold w_{min}.
- **C2** Only the host node can interrogate its associated tags.
- **C3** A host can only interrogate one of its tags at a time.

C1 represents the basic requirement that tags need only one sufficiently strong unmodulated carrier to communicate, given that they would not work well with multiple carriers, as discussed in Section 2. **C2** helps reduce the collision domain of sensor tag communications to a relatively small area around the tag, by ensuring we use a relatively weak signal from a distant node, never the host, as unmodulated carrier. This constraint reflects the dependency of communication range with the strength of the unmodulated carrier as discussed in Section 2. **C3** restricts hosts to interrogate only one tag per cycle to reflect the fact that radios can only receive from one device at a time, and that collisions among backscatter tags occur in the surroundings of the host node and its associated tags. Because of this, co-hosted tags are always interrogated in sequence (e.g. Tags T_2 and T_3 in Figure 4(b)).

To model constraints **C1** to **C3** we demand that every tag must satisfy all three predicates P_1 to P_3 during every cycle when it is being interrogated. This is expressed as:

$$\forall_{t \in T} \forall_{c \in [1, N_t] \colon S_{t,c} = I} P_i(t, c) \colon i \in \{1, 2, 3\} \tag{4}$$

Predicates P_1 to P_3 model constraints **C1**-**C3** respectively and are a function of the tag t being interrogated during cycle c.

We model **C1** by expressing $P_1(t, c)$ in Equation 5. $P_1(t, c)$ represents that there must exist exactly one regular node g set to generate carrier during cycle c and that the signal strength observed by tag t from that carrier generator according to the weight matrix W must exceed a threshold w_{min}:

$$P_1(t, c) \coloneqq \exists!_{g \in A} S_{g,c} = C \wedge W_{g,t} > w_{min} \tag{5}$$

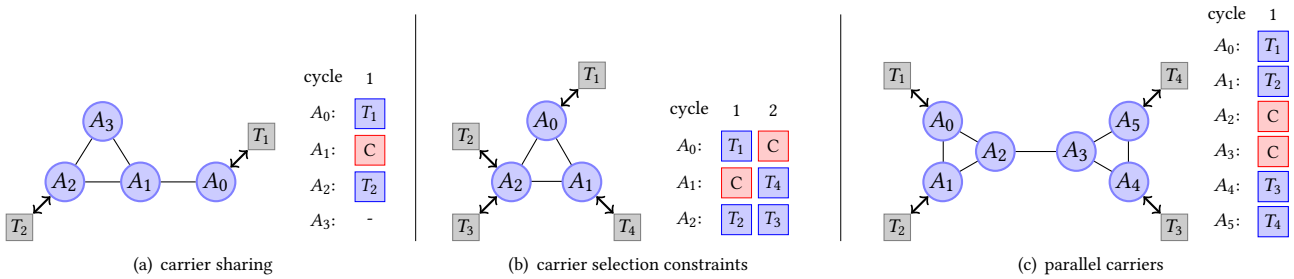
To model **C2** we express predicate $P_2(t, c)$ to require that during cycle c the host H_t of tag t must also be set to interrogate (I) and that tag t and its host must set each other as their destination addresses:

$$P_2(t, c) \coloneqq S_{H_t,c} = I \wedge D_{H_t,c} = t \wedge D_{t,c} = H_t \tag{6}$$

Predicate $P_3(t, c)$ models **C3**. In Equation 7 we demand that every tag k other than t that shares the same host as t must be off (O) during cycle c.

$$P_3(t, c) \coloneqq \forall_{k \in T \colon k \neq t \wedge H_k = H_t} S_{k,c} = O \tag{7}$$

We have omitted additional constraints that only ensure proper modelling and symmetry breaking for efficient computation. These include constraints to model that tags cannot generate an unmodulated carrier or that regular nodes are off when not interrogating a



(a) carrier sharing (b) carrier selection constraints (c) parallel carriers

Figure 4: *Example topologies and corresponding TagAlong schedules.* The examples highlight different aspects of the scheduling process such as opportunities for carrier sharing and parallel carriers.

tag or generating a carrier. For symmetry breaking, which is important to ensure we do not explore redundant solutions, we require that the n-th tag should be interrogated no later than during the n-th non-empty cycle, thus excluding equivalent permutations of the interrogation cycles.

Schedules for a few example topologies are shown in Figure 4.

3.5 Concurrent Carrier Requests

When sharing a carrier generator among multiple tags concurrently, it is possible that carrier requests collide in scenarios as those shown in Figures 4(a) and 4(b). We want multiple nodes to interrogate their assigned tags using a common carrier generator to save energy. In that case, they may all transmit a request for carrier at roughly the same time. These requests are, however, likely to collide and be lost. Consider the example in Figures 4(a). Nodes A0 and A2 interrogate their associated tags at the same time using A1 as unmodulated carrier. If they both transmit requests at the same time, these are likely to be lost.

To solve this issue we note that from the carrier generator's point of view, it is sufficient to know that at least one of its neighbors needs the unmodulated carrier in order to enable it. We adopt the simple strategy depicted in Figure 5 that illustrates how it works for the situation in Figure 4(a): We replace the carrier request message by a short period where the requesting nodes (A_0 and A_2) generate a carrier of their own. During this time, the carrier generating node (A_1) uses its CCA function to detect activity in the channel. Node A_1 will only enable the carrier if it detects channel activity. During this brief period, carrier generating nodes increase their CCA threshold to avoid false positives, increasing robustness. In this scenario, Node A_3 is in range of A_2 while A_2 requests the carrier. However, since node A_3 is not scheduled as carrier generator, it does not need to listen to carrier requests.

Using channel activity to signal carrier generators to engage is a one-bit signal indicating that the carrier should be enabled. This approach does not allow the request to indicate which node should engage the unmodulated carrier. This is the reason for constraint **C1**. It makes sure that every regular node has only one carrier generator nearby during its assigned interrogation cycles. Therefore, there is no possibility of confusion when requesting unmodulated carriers with our solution. For example, due to constraint **C1**, nodes such as A_2 in Figure 4(a) can never have two neighbors assigned as carrier generators during a slot where it is interrogating its tag. In the figure, either A_1 or A_3 could provide the unmodulated carrier, but not both at the same time. In this case, TagAlong selects node A_1 during cycle 1 because it serves to interrogate T_1 at the same time.

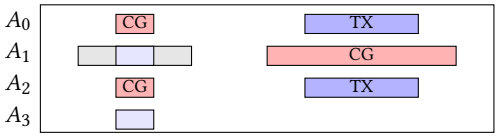


Figure 5: *TagAlong sidesteps possible collisions of carrier requests in the optimized schedule using a brief unmodulated carrier and the CCA mechanism.* Regular nodes A_0 and A_2 can concurrently request a carrier from A_1 by emitting their carrier that A_1 detects because it enables the CCA mechanism (gray). A_3 is not scheduled to generate carrier for this slot.

3.6 Synchronous interrogations

To maximize the energy savings due to carrier reuse, we must ensure interrogations to tags that share the same carrier generator happen in the same slotframe. To that end, TagAlong has a mechanism where it only interrogates tags every n-th globally numbered slotframe, where n is a function of the slotframe duration and the desired interrogation frequency. This gives enough time for neighbors who share a carrier to enqueue interrogation requests from the application and to perform them synchronously.

This mechanism exploits the slotted nature of TagAlong allowing us to keep track of the absolute time. This can be achieved by assigning absolute numbers to timeslots and simply keeping track of the number of elapsed slots.

Most sensing applications where sensor tags are appropriate can afford significant sensing delay, so by default interrogations are performed synchronously. The application layer still has the option to request on-demand interrogations that are performed as soon as possible (during the next slot for the requested tag). In the case of periodic sensing applications, or whenever on-demand interrogations are not needed, TagAlong can avoid listening for carrier generators in every timeslot except for the ones where synchronized interrogations can happen. This can lead to significant energy savings.

3.7 Increasing the Efficiency of the COP

The COP solver explores a large number of combinations in order to determine the optimal schedule. As the size of the network increases, the number of combinations explodes making the problem more computationally intensive. In an effort to alleviate this situation, we introduce a heuristic that reduces the size of the problem the solver needs to tackle without affecting the final solution.

The heuristic builds on the observation that regular nodes that are neither hosts nor neighbors of any tag, play no role in communications with sensor tags and are always off. To simplify the COP we remove these nodes from the graph, reducing the size of the problem. Removing these nodes could, in certain instances, partition the graph into several connected components, which can be solved independently; contributing further to reduce the computational complexity of the problem. The complete tag schedule is built by joining the schedules of all connected components in parallel given that it is guaranteed that there will be no collisions across connected components.

4 IMPLEMENTATION

We have created a proof of concept implementation of TagAlong to show that it is feasible and to use it in our evaluation. The implementation is based on a network of Zolertia Firefly sensor nodes with TI CC2538 radios that are compatible with the IEEE 802.15.4 standard. The regular nodes run the Contiki operating system [6] with its standard IPv6 network stack. As medium access protocol we employ Time-Slotted Channel Hopping (TSCH) [20], which is part of the IEEE 802.15.4 standard. We use RPL [46] as routing protocol.

We created a modified version of TSCH that follows the design of Section 3 in that it adds support for carrier generator slots but it is otherwise compatible with the original implementation. During their carrier generator slots, nodes listen for carrier requests and only enable the carrier, at the appropriate frequency, if a request is received. Conversely, interrogating hosts will emit carrier requests just before interrogating sensor tags according to the design of Section 3. Time is divided into slots that last 10 ms, long enough for a node to receive a frame and transmit an acknowledgement. Each node has its own schedule that dictates what to do during every time slot, whether to receive, transmit, generate a carrier or remain idle. The schedule also indicates the channel on which to receive or transmit during any given timeslot. TSCH nodes typically leverage frequency diversity by hopping over multiple channels.

In our implementation, the regular nodes' schedule can be defined with any TSCH scheduler [8, 28]. This schedule is used, for example, to collect topology information and later to disseminate the schedule for sensor tag interrogations. During our evaluation we employ a static schedule for simplicity.

Our sensor tag prototypes improve on our previous design of an IEEE 802.15.4 transceiver that operates assisted by an external unmodulated carrier [40]. It presents the characteristics described in Section 2: It requires an unmodulated carrier at 8 MHz of the transmitted or received signal and its range depends on the carrier strength. Our prototypes are more sensitive than the original. With an unmodulated carrier strength of −70 dBm, equivalent to a line-of-sight distance of 7 m to the carrier generator, they have a communication range of 25 cm. We set the minimum acceptable carrier signal strength in our COP to be equal to this value ($w_{min} = -70$ dBm).

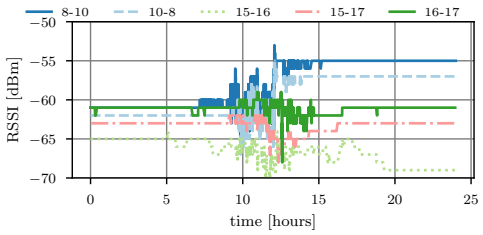


Figure 6: *RPL constantly tracks link state with a smoothing filter that allows our schedule to adapt to significant link variations.* Link state evolution of several carrier generation links in our testbed over 24 hours.

RPL needs reliable and current link-state information to establish routes. Therefore it constantly tracks every link's signal strength with an exponential window moving average filter. We leverage this link state information to discover the topology of the regular nodes' network. Figure 6 shows the evolution over 24 hours of RPL's link quality for several of the links used for carrier generation in our testbed. RPL's filter allows us to ignore rapid RSSI fluctuations while adapting the schedule to meaningful trends when needed.

In our implementation, during topology discovery each node periodically sends its list of neighbors to the cloud server, where it is compiled to generate a full picture of the network. There are many topology discovery mechanism available in the literature that could be adopted [5, 24, 36, 42]. Such a mechanism is beyond the scope of this work. Once a schedule to interrogate tags is computed, it is disseminated to all nodes and it is appended to the regular nodes' schedule as additional time slots. Link state updates may still be sent to the server to update the schedule if the need arises.

In place of the cloud server we employ a desktop computer with an Intel Core i7 CPU at 3.6 GHz and 16 GB of RAM running the Ubuntu operating system. To solve the COP we employ the MiniZinc constraint modelling language [34].

5 EVALUATION

We deploy our implementation from Section 4 in a testbed to show that all the components work together and to illustrate some of the attainable savings, as well as to evaluate its reliability. To increase the scale of our evaluation we then study the savings that we can achieve with the COP by generating instances with random tag allocations in topologies gathered from real research testbeds and on randomly generated ones. We also evaluate the solving time of our COP and the effectiveness of our heuristic.

We make the following key findings:

- We limit the excess latency for communications among regular nodes and provide significant energy savings, without affecting the reliability of tag interrogations.
- Using real-world topologies we show that TagAlong can achieve a 60% reduction in excess latency and a 30% reduction in energy consumption compared to the sequential schedule.

- Our heuristic considerably reduces the worst-time COP solving time. In 96% of the cases we achieve well under 60 s, even for the largest topologies evaluated (24 regular nodes and eight sensor tags).

We deploy our TagAlong implementation from Section 4 in a local sensor network testbed that consists of 25 regular nodes distributed across one floor in an office building. We augment the testbed with six sensor tag prototypes, each located 15 cm from their host node. One node hosts two tags while four other nodes host a single tag each. Every host has at least one neighbor within a distance of 3 m to 5 m.

5.1 Latency

As slotframes become longer due to the addition of tag interrogation timeslots, nodes need to wait longer for their assigned chance to use the medium. In our first experiment, we compare how the latency of communications is affected by the addition of TagAlong's schedule versus the sequential one.

Setup. We configure TagAlong in our testbed to interrogate different sets of tags. We transmit 1000 frames from the regular nodes. We repeat this with TagAlong's schedule and then with the sequential schedule, for each set of tags. Transmissions are requested at random times with an average period of 1.5 s. We measure the transmission latency for regular nodes as the time that passes between the instant when the MAC layer receives the frame from the upper layer until it is actually transmitted.

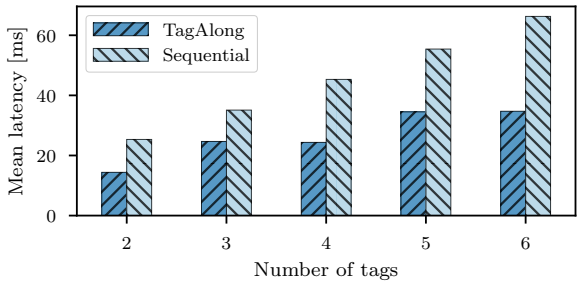
Results. Figure 7 compares the transmission latency between a system with no tags (only regular schedule) and one with tags, using either TagAlong's schedule or the sequential one. Our results show that the added tag slots increase the latency of communications. The expected value for the latency of the sequential schedule is $E(\Delta t_{tx}) = {}^{(N_s \times \tau_s)}/_2$ where N_s is the total length of the slotframe, including the regular schedule (in this experiment $N_s = 2N_t + 1$), and τ_t is the slot duration (10 ms). The results of Figure 7(a) are closely in line with the expected values. TagAlong's schedule scales much better than the sequential one due to its ability to parallelize interrogations. The CDF in Figure 7(b) shows that the latency varies uniformly up to the slotframe duration. This means that most transmissions are fulfilled during the first available slotframe.

The increase in transmission latency that occurs due to the addition of tag slotframes as seen in Figure 7 apply both to communications among regular nodes and tags. In the case of tags, this is only a concern when not using synchronous interrogations. With synchronous interrogations we are choosing to delay tag interrogations up to a certain delay in exchange for higher overall energy efficiency through carrier sharing.

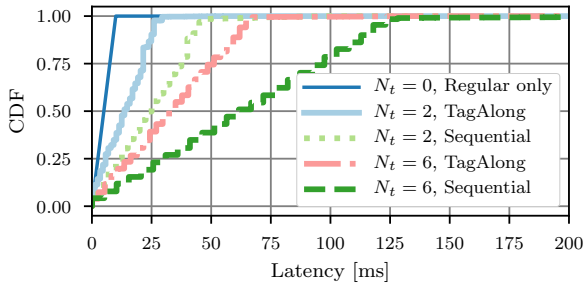
5.2 Energy Consumption

We now illustrate the way that different TagAlong components contribute to energy savings.

Setup. We perform the same experiment as in Section 5.1 with 1000 interrogation cycles for both types of schedules with different sets of tags. We do the experiment with, and without synchronous tag interrogations. We employ Energest [7], the energy estimation mechanism built into the Contiki operating system. In each experiment we estimate the amount of energy the nodes spend



(a) Mean latency among regular nodes.



(b) Cumulative distribution of latency.

Figure 7: *TagAlong maintains low communication latency for the regular nodes as we add more tags.* Adding more tags increases the length of the sequential schedule, leading to increased latency. TagAlong's schedule scales much better as tags are added.

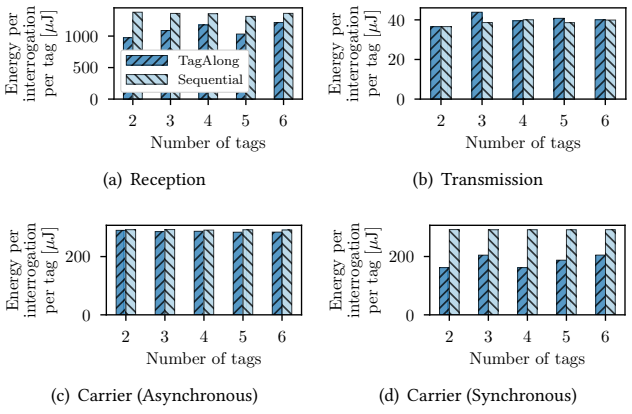


Figure 8: *Most of TagAlong's energy savings due to carrier reuse are only effective with synchronous interrogation.* With asynchronous interrogation TagAlong performs no worse than the sequential schedule in terms of carrier energy consumption.

in transmission, reception or generating carriers during the tag interrogation schedule. Note that the amount of savings depends on the specific topology and tag placement because carrier sharing and parallelization opportunities depend on them.

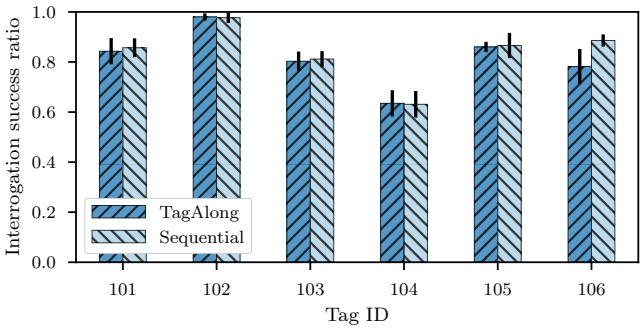


Figure 9: *There is no significant reliability penalty for using TagAlong.* Even though the reliability of tags is different, there is no significant difference between the sequential schedule and TagAlong's. Mean interrogation success ratio over 10 runs of 100 interrogations each, error bars show the standard deviation.

Table 1: *Evaluation Topology Details.* Number of nodes (N_a), average node degree and mean values of carrier and duration ratios for the topologies used in the evaluation. The lower part of the table shows values for the random topologies. For these the average node degree is averaged again over all topologies of the same size and shown with standard deviations.

		Average	Mean η_c		Mean η_d	
Topology	N_a	**Node Degree**	$N_t = 8$	$N_t = 10$	$N_t = 8$	$N_t = 10$
Local	25	9.6	0.65	0.61	0.37	0.35
FlockLab	27	8.4	0.63	0.60	0.31	0.28
D-Cube	39	10.8	0.52	0.49	0.26	0.24
$N_a = 6$	6	3.5 ± 0.7	0.41	0.42	0.39	0.39
$N_a = 12$	12	4.7 ± 1.0	0.44	0.39	0.31	0.27
$N_a = 24$	24	5.5 ± 0.8	0.50	0.44	0.25	0.21
$N_a = 48$	48	6.1 ± 0.6	0.54	0.51	0.18	0.17

Results. Figure 8 shows a comparison of the average energy invested by the regular nodes per tag and per interrogation cycle for each of the radio functions. The results of Figure 8(a) show that for reception, TagAlong invests significantly less energy than sequential interrogation. This is a result of carrier generator sharing, as a lower number of carrier generators have to listen for carrier requests as carrier sharing increases. A large part of the total energy is employed in reception due to the large number of unused slotframes where carrier generators nevertheless have to listen for carrier requests. This overhead can be eliminated almost entirely if we do not need support for on-demand interrogation requests as described in Section 3.6 because in that case, carrier generators only need to listen for requests during active synchronous slotframes.

Figure 8(b) shows that TagAlong invests roughly the same amount of energy for transmissions as the sequential schedule. These values do not change with synchronous interrogations. The energy invested in transmissions is small due to the short length of interrogation frames.

The results in Figures 8(c) shows that in the case of carrier generation without synchronous mode TagAlong invests the same amount of energy as the sequential schedule. This is because there is practically no carrier sharing. Figure 8(d) shows that when we enable synchronous interrogations TagAlong provides significant energy savings by sharing unmodulated carriers. As a comparison, regular nodes spend roughly 147 µJ to transmit a full-length IEEE 802.15.4 frame. In Figures 8(c) and 8(d) we can see that the sequential schedule and TagAlong in the asynchronous mode invests roughly twice the full-frame transmission energy to interrogate a tag, which corresponds to the two carrier generation intervals. TagAlong's schedule requires less energy than the sequential generating carriers.

5.3 Reliability of Tag Interrogations

With the sequential schedule there is no need for compromise in the strength of the unmodulated carrier as we can always select the best neighbor as carrier generator. To show that the reliability of interrogations does not degrade with TagAlong, we compare its interrogation success ratio against sequential interrogation. Figure 9 compares the mean success ratio over 10 runs of 100 interrogations for each case. Error bars represent the standard deviation. The figure shows next to no difference in the reliability of TagAlong compared to the sequential schedule. We assume that the low reliability obtained for some tags, which does not depend on whether we use TagAlong or not, is caused by external interference, as the testbed is located in a busy office building.

5.4 COP Evaluation with Testbed Topologies

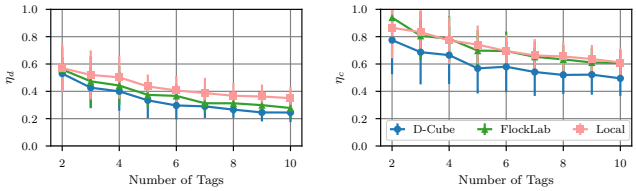
Through our implementation we have shown that our system can provide significant savings in terms of excess latency and energy invested by the regular nodes in generating carriers. However, the efficiency gains obtained with TagAlong depend to a large extent on the specific topology and tag deployment. In the next experiments, we examine the dependency of TagAlong's savings with the size and topology of the network.

In order to perform a larger scale evaluation of our COP, we conduct a set of offline experiments. We use our topology discovery mechanism to collect the topologies of the network of regular nodes from our testbed and from two other open research testbeds: FlockLab [31], and D-Cube [41]. Table 1 shows the number of nodes in each testbed and the average node degree as a measure of the density of these networks.

Our evaluation includes a maximum of 10 tags due to the limited scalability of the COP. The number of regular nodes is much larger to ensure a negligible probability of generating redundant or equivalent random instances in the evaluation.

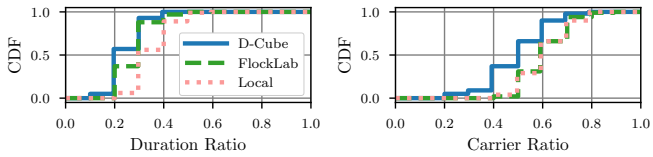
We employ these three topologies as the basis for our experiments, where we create random tag deployments and evaluate two metrics: *carrier ratio* and *duration ratio*.

Carrier ratio ($\eta_c = {}^{n_c}/_{N_t}$) is the fraction of carrier generation cycles in TagAlong's solution (n_c) relative to the number of cycles needed to sequentially interrogating all tags (N_t cycles). Lower values of η_c represent higher carrier reuse and hence more energy and spectrum savings. The carrier ratio is 1 if the number of carrier cycles in the optimized schedule is equal to the one in the sequential schedule (i.e., no optimization is possible). In Figure 4(c), $\eta_c = {}^1/_2$



(a) Duration ratio (b) Carrier ratio

Figure 10: *Our COP significantly improves the efficiency of tag interrogation schedules.* Improvements tend to increase as the number of regular nodes and tags increases. Mean values over 100 realizations for every testbed and N_t combination.



(a) (b)

Figure 11: *Reduction of at least 60% and 30% in duration and carrier ratios in most cases, for the larger N_t's evaluated.* CDFs for three testbed topologies with varying number of nodes (N_a) and $N_t = 10$.

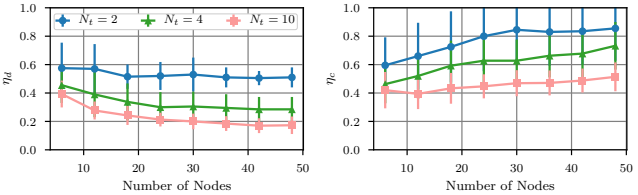
since TagAlong's schedule needs two cycles to interrogate four tags, which would take four cycles to interrogate sequentially.

Duration ratio ($\eta_d = {}^{d_c}/_{N_t}$) is the fraction of the total number of cycles needed to interrogate all tags (d_c) relative to the number needed for sequential interrogation (N_t cycles). Smaller values of η_d mean lower latency. A duration ratio of 1 means no improvement over sequential interrogation. In Figure 4(c), $\eta_d = {}^1/_4$ because TagAlong needs only one timeslot to interrogate four tags.

Setup. For each topology we generate one hundred random tag-to-host assignments of a varying number of tags (N_t) and solve the corresponding COP. For each instance, we compute η_d and η_c.

Results. Figure 10 shows the average behavior of both η_d and η_c as the number of tags increases, for each of the three topologies. The error bars represent the standard deviation. The figures show that the duration ratio improves the most, which translates into an important reduction in latency compared to sequential interrogation. Both of the metrics improve while the number of tags increases as the number of opportunities for parallel interrogations and carrier sharing increase. The carrier ratio figures for D-Cube are slightly better. We attribute this to the high density of this network allowing more carrier reuse. The values obtained for carrier and duration ratios for N_t = 8 and N_t =10 tags, are mentioned in the upper part of the Table 1.

The results of Figure 11 show that in 80% of the cases, with our COP we can decrease the excess latency by more than 60% and the number of carrier slots by 30% relative to sequential interrogation for the larger N_t values tested.



(a) Duration ratio (b) Carrier ratio

Figure 12: *The trends observed on the testbed topologies extends to larger random ones.* The amount of carrier reuse decreases as the density of the network decreases increasing the value of η_c. Examples for one hundred runs of random geometric graphs.

5.5 COP Evaluation with Large Scale Random Topologies

The experiments of Section 5.4 show promising results in reducing the duration of the tags' schedule and the number of carrier generation slots. However they are limited in that they evaluate only three specific network topologies. To evaluate TagAlong with a wider set of cases we perform more offline experiments as in Section 5.4 but with entirely random topologies.

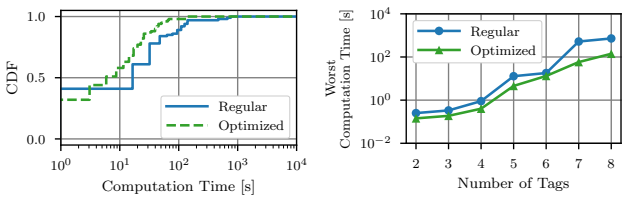
Setup. We generate one hundred random network topologies with varying numbers of regular nodes (N_a). For each of them, we then generate a random tag-to-host assignment with N_t tags. To generate the network of regular nodes, we employ random geometric graphs [38] as a stand-in for a realistic network topology [27]. The lower section of Table 1 summarizes the average characteristics of these topologies, that are average node degree and mean values of carrier and duration ratios for N_t = 8 and N_t =10 tags. To provide the weight of the edges, which represents the signal strength observable through every link, we use the Friis Equation [3]. As in Section 5.4, we compute η_d and η_c for each instance.

Results. Figure 12 shows the behavior of η_d and η_c as we vary the number of regular nodes between six and 48 for different numbers of tags. The results show that we retain the general trend observed with the testbed topologies in Section 5.4. There appears to be only a slight tendency for η_d to decrease with the size of the network (N_a). However, η_c appears to increase slightly for larger networks. We believe that this happens because, as the network size increases, it becomes less likely that a node will have multiple neighboring tags to concurrently serve as carrier generator. This reduces the likelihood of carrier sharing, which increases the number of necessary carrier generation slots and consequently, the value of η_c. On the other hand, a lower tag density makes it increasingly easy to find ways to interrogate tags concurrently using different carrier generators, which explains the downward trend of η_d.

5.6 COP Solving Time

In some instances the COP can take a long time to solve. With the next experiment we want to show that in most cases we obtain a solution in reasonable time. We also investigate the effect of the heuristic introduced in Section 3.7.

Setup. We compare the computation time needed to optimize the 100 instances used in Section 5.5 with (Optimized) and without



(a) $N_a = 24, N_t = 8$ (b) $N_a = 24$

Figure 13: *In most cases our COP solves in under* 60 s. Examples for 24 regular nodes. As seen in Figure 13(a) 96% of the cases are computed in 60 s or less. Our optimization heuristic reduces the worst computation time significantly in many cases.

(Regular) applying the heuristic of Section 3.7 that removes nodes that are not neighbors of any tag.

Result. Figure 13 shows the results. Figure 13(a) depicts that 96% of the cases are computed in 60 s or less in the case of eight tags in networks of 24 regular nodes. The computation time appears to asymptotically scale exponentially with the number of tags, as indicated by the linearly-growing curve in the semi-log plot of Figure 13(b). The optimization reduces the worst computation time by nearly an order of magnitude in some cases. It is to be expected, however, that the heuristic of Section 3.7 does not provide a significant improvement in networks with a large density of tags. This is because the regular nodes are more likely to be connected to a tag as we install more in the network, rendering the heuristic ineffective as we can no longer remove many regular nodes.

6 RELATED WORK

Optimizing TDMA schedules has been an active area of research in conventional wireless sensor networks. Ergen et al. [11] and Gandham et al. [13] have used optimization to determine the smallest conflict-free slot assignment. Zhang et al. have addressed the problem of joint link scheduling and channel assignment for WirelessHART [48]. Similar to these studies, we also use optimization. Our approach, however, differs in that sensor tags require an external carrier to operate and they can have very reduced communication range compared to a normal sensor node. This leads to fundamentally different rules for what is considered a valid configuration and therefore results in different optimization problems than those involving only regular radio links. Specifically, this allows optimizing carrier allocation by reusing a single carrier for multiple concurrent interrogations.

Our work is related to battery-free communications that employ backscatter and other related techniques to achieve ultra-low power consumption, in particular to those who integrate battery-free communications with commodity networks. Most of the work in this area focuses on the physical layer but the efficient provision of the excitation signal is rarely addressed. By contrast, in our work, we focus on providing the excitation signal in the most efficient way to cause as little disruption as possible to the operation of the regular network nodes and other nearby devices.

We previously introduced the idea of augmenting an IoT network with sensor tags and having the regular nodes provide the unmodulated carrier [39]. That work, however, did not address carrier scheduling mechanisms. In subsequent work [40], we introduced two-way ultra-low power communication and scheduled the carrier statically in a TDMA protocol, very similar to our current implementation. However, we demonstrated the concept with a single prototype, without optimizing carrier generators in any way. This paper builds on our previous work, but our goal now is to provide the unmodulated carrier in the most efficient way and to cause little disruption to regular nodes in a networks with multiple deployed sensor tags. We also demonstrate the system in a small scale testbed.

Netscatter [16] integrates battery-free devices into commodity networks and leverages specific properties of LoRa's physical layer to decode multiple concurrent transmissions at a specialized base station. TagAlong differs in that it does not assume a specific physical layer, and in that it extends a mesh network of unmodified standard IoT devices with battery-free tags. Several other works integrate battery-free devices in standard networks [9, 21, 26, 43, 44]. These works make varying efforts to make efficient use of the unmodulated carrier. In some cases the carrier is always on, while in others its duration is tailored to the known duration of backscatter transmissions. In none of these cases, however, the authors share carriers between devices or limit the disruption of standard networks like TagAlong does.

A different approach is taken by works that employ modulated traffic as the excitation signal [25, 37, 47, 49]. While these approaches leverage an information-carrying signal as excitation, they have less control over it. As a result these works either make no attempt to optimize the carrier or sometimes try to make the regular network perform spurious transmissions.

Braidio [18] takes a radical new approach by dynamically switching the task of carrier generation between the transmitter and the receiver. Braidio focuses on maximizing the system lifetime but does not attempt to optimize carrier usage other than within a single radio link.

Gummeson et al. [14] address the link layer for RFID-like devices. Their focus is on increasing the data rate of bulk transmissions while remaining compatible with EPC Gen 2 RFID readers. Instead, we focus on efficient use of the unmodulated carrier.

Van Huynh et al. [19] employ numerical analysis to optimize the overall network throughput in a network of RFID devices powered through RF. While they focus on optimizing the energy harvesting of their tags, we do not assume, or rule out, any harvesting modality. Decoupling energy harvesting from communications allows us to directly interoperate with the standard networks while remaining independent of the harvesting modality. Li et al. [30] propose a polling protocol for the same kind of system. They focus on polling as many tags as possible with conventional RFID readers, therefore they are not concerned with saving energy.

7 DISCUSSION

Dynamic environment. Wireless links can vary broadly in time. Our system constantly monitors link quality so it can react to these variations. It can, for instance, update the schedule if existing nodes die or new ones are added. Furthermore, the system can update the schedule in reaction to significant link quality changes while ignoring spurious fluctuations by filtering, as discussed in Section 4.

In our implementation these updates would be necessary roughly as often as RPL updates a preferred parent, which is known to occur seldom in dense networks [12].

High Network Density. Current sensor tag technology has limited communication range, therefore it requires fairly dense networks of regular nodes to operate (5 m to 7 m inter-node distance). However, tag technology is advancing rapidly and recent works have shown a lot of promise in increasing the communication range for these devices [44, 45]. These advances can help reduce TagAlong's limitations in both network density and tag range.

Centralized Approach. TagAlong adopts a centralized approach which limits scalability due to the high computational cost of the schedule computation, along with the need to collect the topology at a single point and the requirement for cloud access. TagAlong would benefit greatly from a distributed scheduling algorithm because that would solve most of these issues. Note that the remaining components of TagAlong would still work the same way. We leave the design of a more capable scheduler for future work.

8 CONCLUSIONS

We introduced TagAlong that, to the best of our knowledge, is the first system to efficiently coordinate carrier generation in an IoT network augmented with interoperable battery-free tags. Our testbed-based evaluation shows that TagAlong minimizes the disruption to the operation of regular nodes in terms of added latency and power consumption. TagAlong also avoids unnecessarily polluting the RF spectrum with unmodulated carriers. Finally, TagAlong achieves all this with no discernible decrease in reliability in communications to sensor tags when compared to the alternative approach of interrogating each tag in sequence.

ACKNOWLEDGEMENTS

Work partly funded by the Swedish Research Council (Grants 2017-045989 and 2018-05480) and a Google Faculty Research Award.

REFERENCES

[1] 2014. Firmware repository for the WISP 5.0. https://github.com/wisp/wisp5
[2] 2019. UMich Moo: A Programmable RFID-Scale Sensor Device. https://spqr.eecs.umich.edu/moo/
[3] C. A. Balanis. 2005. *Antenna Theory: Analysis and Design.* Wiley-Interscience.
[4] N. Bhatti et al. 2016. Energy Harvesting and Wireless Transfer in Sensor Network Applications: Concepts and Experiences. *ACM Trans. Sen. Netw.* 12, 3 (2016), 24:1–24:40.
[5] S. Devasenapathy et al. 2013. Between Neighbors: Neighbor Discovery Analysis in EH-IoTs. In *10th International Conference on Autonomic Computing (ICAC '13).*
[6] A. Dunkels et al. 2004. Contiki - A Lightweight and Flexible Operating System for Tiny Networked Sensors. In *29th Annual IEEE International Conference on Local Computer Networks. (LCN'04).* IEEE, 455–462.
[7] A. Dunkels et al. 2007. Software-based On-line Energy Estimation for Sensor Nodes. In *4th Workshop on Embedded Networked Sensors (EmNets '07).* ACM.
[8] S. Duquennoy et al. 2015. Orchestra: Robust Mesh Networks Through Autonomously Scheduled TSCH. In *SenSys'15.* ACM.
[9] J. Ensworth et al. 2015. Every smart phone is a backscatter reader: Modulated backscatter compatibility with Bluetooth 4.0 Low Energy (BLE) devices. In *IEEE RFID 2015.*
[10] J. Ensworth et al. 2017. A low power 2.4 GHz superheterodyne receiver architecture with external LO for wirelessly powered backscatter tags and sensors. In *IEEE RFID 2017.*
[11] S. C. Ergen and P. Varaiya. 2010. TDMA scheduling algorithms for wireless sensor networks. *Wireless Networks* 16, 4 (2010), 985–997.
[12] J. Eriksson et al. 2018. Scaling RPL to Dense and Large Networks with Constrained Memory. In *EWSN 2018.*
[13] S. Gandham et al. 2008. Distributed time-optimal scheduling for convergecast in wireless sensor networks. *Computer Networks* 52, 3 (2008).
[14] J. Gummeson et al. 2012. Flit: A bulk transmission protocol for RFID-scale sensors. In *MobiSys '12.* ACM, 71–84.
[15] J. Gummeson et al. 2017. RFID light bulb: Enabling ubiquitous deployment of interactive RFID systems. *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies* 1, 2 (2017), 12.
[16] M. Hessar, A. Najafi, and S. Gollakota. 2018. NetScatter: Enabling Large-Scale Backscatter Networks. In *NSDI 2018.*
[17] J. Hester and J. Sorber. 2017. The Future of Sensing is Batteryless, Intermittent, and Awesome *(SenSys '17).* ACM, Delft, Netherlands, 21:1–21:6.
[18] P. Hu et al. 2016. Braidio: An Integrated Active-Passive Radio for Mobile Devices with Asymmetric Energy Budgets *(SIGCOMM '16).* ACM, 384–397.
[19] N. V. Huynh et al. 2018. Optimal Time Scheduling for Wireless-Powered Backscatter Communication Networks. *IEEE Wireless Communications Letters* 7 (2018), 820–823.
[20] IEEE. 2015. *IEEE Standard for Low-Rate Wireless Networks.* 709 pages.
[21] V. Iyer et al. 2016. Inter-Technology Backscatter: Towards Internet Connectivity for Implanted Devices *(SIGCOMM '16).* ACM, 356–369.
[22] N. Jackson et al. 2019. Capacity over Capacitance for Reliable Energy Harvesting Sensors. In *IPSN 2019.*
[23] C. Jiang et al. 2019. 3D-OmniTrack: 3D tracking with COTS RFID systems. In *IPSN '19.* ACM, 25–36.
[24] N. Karowski, K. Miller, and A. Wolisz. 2018. Greedy Multi-Channel Neighbor Discovery. *CoRR* abs/1807.05220 (2018).
[25] B. Kellogg et al. 2014. Wi-fi Backscatter: Internet Connectivity for RF-powered Devices *(SIGCOMM '14).* ACM.
[26] B. Kellogg et al. 2016. Passive Wi-Fi: Bringing Low Power to Wi-Fi Transmissions *(NSDI '16).* 151–164.
[27] H. Kenniche and V. Ravelomananana. 2010. Random Geometric Graphs as model of Wireless Sensor Networks. In *ICCAE 2010*, Vol. 4. 103–107.
[28] S. Kim, H. Kim, and C. Kim. 2019. ALICE: Autonomous Link-based Cell Scheduling for TSCH. In *(IPSN'19).* 121–132.
[29] M. Kotaru et al. 2017. Localizing Low-power Backscatter Tags Using Commodity WiFi *(CoNEXT '17).* ACM.
[30] B. Li, Y. He, W. Liu, and L. Wang. 2019. Towards time-efficient localized polling for large-scale RFID systems. *Computer Networks* 150 (2019), 250–262.
[31] R. Lim et al. 2013. FlockLab: A Testbed for Distributed, Synchronized Tracing and Profiling of Wireless Embedded Systems *(IPSN '13).* ACM, 153–166.
[32] V. Liu et al. 2013. Ambient Backscatter: Wireless Communication out of Thin Air *(SIGCOMM '13).* ACM, 39–50.
[33] R. Nandakumar et al. 2018. 3D Localization for Sub-Centimeter Sized Devices. In *ACM SenSys 2018.* ACM, 108–119.
[34] N. Nethercote et al. 2007. MiniZinc: Towards a Standard CP Modelling Language. In *Principles and Practice of Constraint Programming – CP 2007 (Lecture Notes in Computer Science)*, Christian Bessière (Ed.). Springer Berlin Heidelberg, 529–543.
[35] P. Pannuto et al. 2018. Slocalization: Sub-uW Ultra Wideband Backscatter Localization *(IPSN '18).* IEEE.
[36] G. Papadopoulos et al. 2016. Low-power Neighbor Discovery for Mobility-aware Wireless Sensor Networks. *Ad Hoc Netw.* 48, C (Sept. 2016), 66–79.
[37] Y. Peng et al. 2018. PLoRa: A Passive Long-range Data Network from Ambient LoRa Transmissions *(SIGCOMM '18).* ACM, 14.
[38] M. Penrose. 2003. *Random Geometric Graphs* (1 ed.). Oxford University Press.
[39] C. Pérez-Penichet et al. 2016. Augmenting IoT Networks with Backscatter-enabled Passive Sensor Tags. In *Proceedings of the 3rd Workshop on Hot Topics in Wireless (HotWireless '16).* 23–27.
[40] C. Pérez-Penichet et al. 2018. Battery-free 802.15.4 Receiver *(IPSN '18).* IEEE, 164–175.
[41] M. Schuß et al. 2017. A Competition to Push the Dependability of Low-Power Wireless Protocols to the Edge *(EWSN '17).* Junction Publishing, USA.
[42] M. Seliem et al. 2014. Performance evaluation and optimization of neighbor discovery implementation over Contiki OS. *2014 IEEE World Forum on Internet of Things, WF-IoT 2014*, 119–123.
[43] V. Talla et al. 2017. LoRa Backscatter: Enabling The Vision of Ubiquitous Connectivity. *Proc. ACM Interact. Mob. Wearable Ubiquitous Technol.* 1, 3.
[44] A. Varshney et al. 2017. LoRea: A Backscatter architecture that achieves a long communication range. In *ACM SenSys 2017.* ACM Digital Library.
[45] A. Varshney et al. 2019. TunnelScatter: Low Power Communication for Sensor Tags Using Tunnel Diodes *(MobiCom '19).* ACM, 50:1–50:17.
[46] T. Winter (ed.). 2012. RPL: IPv6 Routing Protocol for Low-Power and Lossy Networks. https://tools.ietf.org/html/rfc6550
[47] P. Zhang, D. Bharadia, K. Joshi, and S. Katti. 2016. HitchHike: Practical Backscatter Using Commodity WiFi *(SenSys '16).* ACM, 259–271.
[48] H. Zhang et al. 2009. Optimal link scheduling and channel assignment for convergecast in linear WirelessHART networks. In *International Symposium on Modeling and Optimization in Mobile, Ad Hoc, and Wireless Networks.* IEEE, 1–8.
[49] P. Zhang et al. 2017. FreeRider: Backscatter Communication Using Commodity Radios *(CoNEXT '17).* ACM, Incheon, Republic of Korea.