Instant: A TSCH Schedule for Data Collection from Mobile Nodes

Atis Elsts[1,2], James Pope[3], Xenofon Fafoutis[4,2],
Robert Piechocki[2], and George Oikonomou[2]
[1]Institute of Electronics and Computer Science, Latvia
[2]University of Bristol, UK
[3]University of Montevallo, USA
[4]Technical University of Denmark, Denmark
atis.elsts@edi.lv

Abstract

Low-power wearable devices are becoming increasingly important for fitness and healthcare applications. However, existing protocols based on the IEEE 802.15.4 low-power wireless standard are not optimized for data collection from mobile devices. This paper presents *Instant*: a schedule for the IEEE 802.15.4 TSCH protocol tailored for this application. We evaluate the data collection speed, energy consumption, and fairness of *Instant*, and show that *Instant* achieves several times higher data collection speed from mobile nodes compared with the state-of-the-art Orchestra schedule.

Categories and Subject Descriptors

C.2.2 [**Networks**]: Network Protocols

General Terms

Algorithms, Performance

Keywords

TSCH, Wearables, Mobility

1 Introduction

Wearable devices play a key role as an enabling technology for the next generation Internet of Things (IoT), including in fitness and healthcare applications [38], such as smart home, smart gym, and smart hospital applications. The recently standardized IEEE 802.15.4 TSCH (Time Slotted Channel Hopping) protocol [4] similarly aims to enable new applications in low-power wireless networks, with focus on those with high reliability and low latency requirements. Even though the IETF 6TiSCH (*IPv6 over the TSCH*) working group [3] has made good progress in standardizing a complete TSCH network stack, TSCH networks that include wearable devices remains a little-explored topic.

This work aims to close this gap and bridge these two technologies: wearables and TSCH. Our focus is on applications in which mobile wearable devices are used together with stationary infrastructure devices (*access points* further in the paper) to provide both indoor localization and data collection services, such as in the SPHERE application (a Sensor Platform for Healthcare in a Residential Environment [13]). These applications typically use periodic beacons for localization in conjunction with unicast traffic for data collection from mobile nodes. They create a number of challenges for state-of-the-art TSCH networks:

- **Neighbor discovery.** The unicast probing mechanisms of standard IoT routing protocols such as RPL are not optimal for networks with mobile nodes: the set of neighbors to probe can become stale every time these nodes move. Instead, some sort of anycast is preferable.
- **Anycast.** However, there are no provisions for efficient and reliable anycast in TSCH and 6TiSCH standards.
- **Cell reservations.** The mechanisms for scheduling TSCH cells (*e.g.*, the IETF 6top protocol [34]) are not designed for mobile networks; for example, there is no functionality to automatically free cells reserved by a node that has since left the transmission range.
- **Fairness.** Simple strategies – such as allowing the bandwidth of a single access point to be completely reserved by a single wearable – lead to starvation of wearables (*i.e.*, having no reserved cells to any access point) if there are more wearables than AP in a range.

The main contribution of this work is *Instant*, a TSCH schedule for fast, reliable, and fair data collection from mobile nodes[1]. *Instant* works as follows: the mobile nodes periodically probe the access points with broadcast packets. These probing packets are received and acknowledged by all access points in the radio reception range. The acknowledgments (ACK) to these probing packets include information about reservations of subsequent unicast cell blocks. If a mobile node wishes to transfer high-rate or bulk data to the infrastructure network, for example, to upload sensor data saved in a local storage device while being out of the range of the network, it uses the information from these ACK to select which access point to use as the next hop for the data transmissions. Many wearable applications already use transmissions of periodic broadcast packets (for instance, to implement indoor

[1]Source code available: `https://tinyurl.com/ybkyh89v`.

International Conference on Embedded Wireless
Systems and Networks (EWSN) 2019
25–27 February, Beijing, China
© 2019 Copyright is held by the authors.
Permission is granted for indexing in the ACM Digital Library
ISBN: 978-0-9949886-3-8

localization [8, 10, 25]), these applications can apply *Instant* scheduling with minimal overhead; for them, the only extra energy consumption in the probing step comes from the need to keep the radio on to receive the ACK.

This paper makes these specific contributions:

- Comparison of multiple reliable anycast ACK techniques in TSCH. We experimentally show that it is possible to reliably receive acknowledgments from multiple nodes within a standard-length TSCH timeslot.
- Design of *Instant*: a novel unicast slot allocation technique for networks with mobile nodes.
- Quantification of *Instant* throughput and energy consumption in static, single-wearable scenarios, and its comparison with Bluetooth Low Energy (BLE): a leading low-power wireless protocol for wearables. We show that *Instant* provides application-layer performance that is comparable with BLE 4 Connected Mode [7] on the same hardware, despite the latter being specifically focused on static, point-to-point operation, and despite the lower energy efficiency [29] and datarate [4] of the IEEE 802.15.4 physical layer.
- Quantification of *Instant* properties in scenarios with multiple wearables, both mobile and static. We show that for mobile nodes, *Instant* provides significantly better data collection speed and fairness than the state-of-the-art RPL-with-Orchestra network stack [11], and at the same time shows performance similar to RPL with Orchestra in static scenarios.

2 Related Work

The TSCH protocol is a Time Division Multiple Access (TDMA) Media Access Control (MAC) protocol for low-power wireless networks. It is defined in the IEEE 802.15.4-2015 standard [4]. The main uses cases of TSCH to this date are in industrial monitoring [35]. However, it has reliability and predictability properties that make it appealing for other applications, including applications in healthcare and fitness domains. The IEEE 802.15.4 standard does not define a specific schedule for TSCH, but multiple approaches have emerged from the research community and the standard organizations. Orchestra [11] is an autonomous scheduling mechanism that uses the RPL routing state to decide which slots to activate on each node. The IETF 6TiSCH (*IPv6 over the TSCH mode of IEEE 802.15.4e*) Working Group has defined the 6top protocol for distributed scheduling [34] and is working on the 6TiSCH Minimal Scheduling Function [9].

Neither Orchestra nor 6top are designed to support mobile nodes: they do not consider the problems of discovery and handover of access points. Huynh *et al.* [19] optimize TSCH by allowing to send packets to multiple parents nodes opportunistically, which is an approach similar to *Instant*. However, they do not use the reliable anycast ACK technique, and their approach is neither targeted for, nor evaluated in mobility scenarios. Haxhibeqiri *et al.* [17] focus on the problem of mobility on top of TSCH by allowing mobile nodes to roam between multiple infrastructure nodes. However, they only support low-rate, broadcast traffic in the upstream direction, in contrast to *Instant*. Al-Nidawi *et al.* [5] provide a mobility framework with the main focus on improving the association time in TSCH. In contrast, our work focuses on the operation after the association has taken place, and could be used in conjunction with their approach.

BLE [7] is a prevalent low-power standard that traces its roots to IEEE 802.15.1. BLE is designed for short-range cable-replacement applications in star-type topologies. Despite the fact that BLE is very commonly used for mobile applications, such as wearable sensors [27], the standard does not support mobility. Instead, in most practical applications, BLE mobility is implemented indirectly through smartphones [26, 21, 39]. The main idea of this approach is that the smart-phone is most of the times in close proximity to the wearable sensor. In other words, a static body-to-body link is established that allows the user to be mobile. This approach is adopted by the majority of commercial wearable devices, such as smart watches and fitness trackers [24].

Whilst the BLE standard itself does not support mobility, extensions to this end can be found in the literature. These works follow two different approaches, namely: handovers and broadcasting. With the handover approach, mobility is implemented by the peripheral establishing a new connection to a different BLE master, seamless to the application layer. Some early works that introduce handover support in classic Bluetooth can be found in [6] and [22]. In [32], the authors propose Mobile Hub: a middleware solution to support mobility across different smartphones. In [18], the authors propose SeamBlue, which implements seamless connection migration in BLE. The key disadvantage of this approach is that it introduces a handover overhead that is associated to the energy consumed for establishing a BLE connection. This overhead is impractical if the mobility patterns of the user require frequent handovers. The second approach is based on broadcasting BLE advertisements. In [14], the authors present a system that implements BLE mobility by communicating data over non-connectable undirected advertisements. Each advertisement packet is retransmitted once times on each of the BLE broadcast channels, since the receivers are not synchronized with the transmitters. This approach lacks reliability, as broadcast transmissions are not acknowledged, and has severely limited throughput: according to the standard, advertisements can carry up to 24 bytes of data and be transmitted at a maximum frequency of 10 Hz. Contrary to these works, *Instant* supports reliable mobility without limiting the throughput and without introducing significant connection establishment overheads.

There are other options that support mobility in multi-hop wireless networks, for example, the Low-Power Wireless Bus [15] (LWB). However, LWB requires platform-specific implementation and is not supported on CC2650 hardware due to its lack of the start-of-frame-descriptor interrupt.

3 Design of *Instant*

3.1 Overview

Conceptually, the algorithm must solve three problems:

- **Neighbor discovery.** Both wearables and access points must have an up-to-date view of their neighbors.
- **Cell allocation.** In a neighborhood with n wearables and m access points, $n \times m$ links are possible in total;

Table 1. Notation reference

Name	Definition
$\tau_{slotframe}$	Slotframe duration
τ_s	Slot duration
τ_t	Transmission offset
τ_p	Packet duration
τ_i	Idle period between packet and the first ACK
τ_a	ACK duration
N_a	Number of ACK subslots
$Addr_{node}$	The address of *node*
$Addr_a$	The anycast address
$O_{probing}$	The channel offset of probing cells
O_{node}	The channel offset of *node*
T_{fresh}	Max time to remember a wearable as present
$a_{slotframes}$	Number of active slotframes for a wearable

however, just $min(m,n,k)$ can be active simultaneously (where k is the number of TSCH channels). The wearables and the access points must agree on which links to activate and reflect that in their schedules.

- **Cell deallocation.** When a wearable either leaves the neighborhood or has transmitted all of its data, the cells allocated for it must be freed.

Another desirable property of such an algorithm is good performance during periods when the wearables are either stationary, or mobile, but within the range of just a single access point (AP). This is important because in smart-home applications, no more than a single AP per room is typically deployed, and movement of the inhabitants between different rooms is an exception, rather than a norm [33].

In order to implement **neighbor discovery** we assume an application that uses periodic beacons for indoor localization, and leverage this periodic traffic.

The design of the **cell allocation** algorithm in *Instant* is influenced by process scheduling in operating systems (*e.g.*, the Completely Fair Scheduler in Linux [36]); the concept of a *timeslice* used in these schedulers corresponds to a slotframe in *Instant*. The Linux scheduler avoids starvation of processes by providing a timeslice of the CPU to all waiting processes in a round-robin fashion. The access points in *Instant* similarly provide "slices" of their future schedule to wearables within the communication range that have data to send; this achieves fairness (in the narrowly defined sense of "avoiding starvation", which is our design goal). The wearables are selected in a round-robin, random or another fashion depending on the implementation (see Section 3.3 for details). However, there is a crucial difference between CPU scheduling and TSCH scheduling: coordinated decision making is vastly more difficult for the latter, since communication in low-power wireless networks is much more expensive and less reliable. For this reason, *Instant* avoids both direct communication between different AP and direct communication between different wearables.

Cell deallocation is a trivial problem in *Instant* due to its timeslice-based nature: if a slotframe is not explicitly allocated for a wearable, it is considered free. If multiple slotframes have been allocated at once for a wearable, the cells

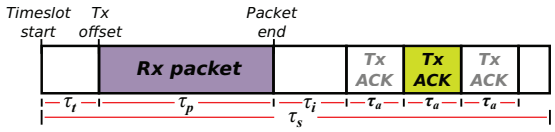


Figure 1. A TSCH slot with $N_a = 3$ ACK subslots.

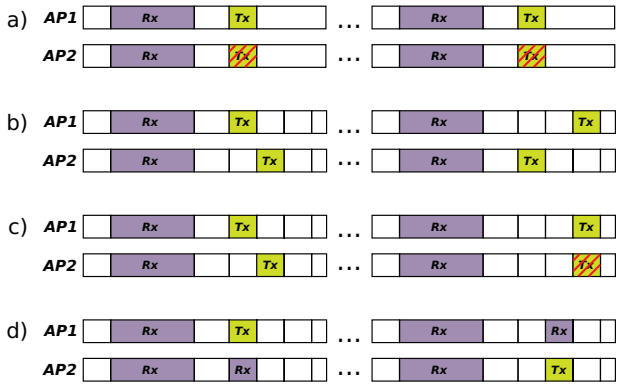


Figure 2. Comparison of the different anycast ACK techniques, with $N_a = 3$. a) Synchronous ACKs. b) Pseudo-random ACKs, no collisions. c) P-random ACKs with light collisions. d) P-random ACKs with overhearing, no collisions. ACK lost due to Tx collisions shown red.

in all subsequent slotframes are deallocated after an "empty" slotframe, in which no data was received from that wearable.

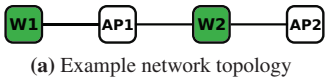
Finally, the light-weight nature of cell allocation and deallocation in *Instant* means that *Instant* is also suitable for static networks. In completely stationary conditions, allocating larger blocks of cells is preferable, since increasing the size of the allocation reduces the number of control messages that need to be sent. For this reason, an *Instant* access point dynamically tunes the size of its allocations depending on the stability in the neighborhood (*i.e.*, in the set of its neighboring wearable devices). Allocating an unbounded number of cells turns *Instant* into a connection-oriented protocol.

3.2 Reliable Anycast

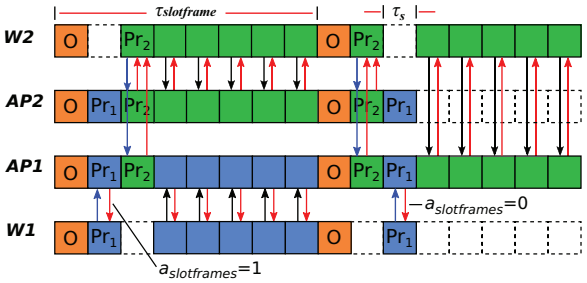
Instant wearables send out anycast probing packets to discover access points in the neighborhood. All AP statically schedule cells for listening to these probing messages. Such a cell is uniquely identified by its slot offset S_W and channel offset $O_{probing}$. Upon receiving the probing packet, the AP decides whether to allocate the subsequent unicast slots for the wearable, and sends an ACK to inform the wearable of the result.

However, this approach presents the problem of ACK collisions. One solution for the ACK collisions is to rely on the strong capture effect present in IEEE 802.15.4 networks. For this, the ACK should be sent synchronously, with less than 128 μs timing difference [37] (Fig. 2a). The wearable is going to receive the ACK with the strongest signal strength as long as the sum of the other signals is at least 3 dB lower [16]. The main drawback is that the capture effect does not scale well for a large number of simultaneous transmissions.

Another solution is to distribute ACKs in time. Each ACK could be sent in its own timeslot; however, that would re-



(a) Example network topology



(b) Corresponding *Instant* slotframes

Figure 3. The *Instant* scheduling approach for an example network. $W1$, $W2$: wearables, $AP1$, $AP2$: access points. In the first slotframe, $AP1$ selects the wearable $W1$ and $AP2$ selects $W2$. In the second, $AP1$ selects $W2$ and $W1$ is left without an access point to use. The different colors denote different channel offsets (c.f. Fig 9). Pr: probing cells for negotiations; O: cells for other traffic, e.g., for RPL traffic between access points and access-point-to-wearable communication. Other cells are idle.

duce the total capacity of the schedule. We observe that it is possible to fit multiple ACK in a standard-length TSCH timeslot (Fig.1). Let us follow the notation in Table 1. The number of ACK that can be sent in a TSCH slot is given by:

$$N_a = \left\lfloor \frac{\tau_s - (\tau_t + \tau_p + \tau_i)}{\tau_a} \right\rfloor. \quad (1)$$

It takes $32\,\mu$s by a IEEE 802.15.4 radio to transmit a byte, and the maximum IEEE 802.15.4 packet size after the synchronization header is 128 bytes, therefore $\tau_p \leq 128 \times 32 = 4096\,\mu$s. Assuming TSCH standard values $\tau_s = 10000\,\mu$s, $\tau_t = 2100\,\mu$s (note that τ_t refers to the *end* of the synchronization header, not the start) and experimentally verified[2] as sufficient $\tau_i = 1000\,\mu$s, that leaves $2804\,\mu$s for the ACK. The size of an ACK is generally much smaller than the maximal frame size; assuming 11 byte ACK, we have experimentally verified that $800\,\mu$s is sufficient for τ_a. Consequently, $N_a = 3$ for standard TSCH timing, which can be further increased if the maximal packet and ACK sizes or processing times are reduced, or τ_t shifted.

A dedicated ACK subslot can be pseudorandomly allocated to each AP with a hash function $h(AP,ASN)$, where the ASN is the Absolute Sequence Number of the slot (Fig. 2b). In this paper, we use $ACK_{offset} = (AP + ASN)$ **mod** A_n as the function h. In small networks, this function is collision free. For larger networks, some ACK collisions are expected, as there are more AP than there are ACK subslots (Fig. 2c), however, as long as the expected number of collisions is small, it is efficient to exploit the capture effect.

One more variation of the pseudorandom ACK technique involves sniffing for ACK from other AP and going back to sleep without sending own ACK if an ACK from another AP is received (Fig. 2d). This allows to reduce the number of ACK sent, potentially reducing collisions and saving energy.

[2]Here and further: on TI CC2650 System-on-Chip hardware.

3.3 Cell Allocation Algorithm

Algorithm 1 The Instant algorithm: wearable operation

```
AP_s ← NULL                                   ▷ The selected access point
a_slotframes ← 0                              ▷ Number of slotframes still selected
n_pkt ← 0                                     ▷ Number of packets ACKed in last slotframe
data ← QUEUE(sensorData)                      ▷ Data packet queue
p ← NULL                                      ▷ Probing or data packet
ack ← NULL                                    ▷ Acknowledgment packet

function ONPROBINGTIMESLOT( )                 ▷ Probing for localization
    p ← CONSTRUCTEMPTYPACKET( )
    p.queuesize ← len(data)
    TRANSMITPACKET(p, Addr_a, O_probing)
    if a_slotframes = 0 then                  ▷ Need to re-select AP?
        AP_candidates ← ∅
        for i ∈ [1, N_a] do
            ⟨ack, rssi⟩ ← RECEIVEACK( )
            if ack ≠ NULL AND ack.a_slotframe ≠ 0 then
                APPEND(AP_candidates, ⟨ack, rssi⟩)
            end if
        end for
        ⟨AP_s, a_slotframes⟩ ← SELECTAP(AP_candidates, AP_s)
    end if
end function

function ONUNICASTTIMESLOT( )                 ▷ Data collection
    if AP_s ≠ NULL AND len(data) > 0 then
        TRANSMITPACKET(data.front( ), Addr_AP_s, O_self)
        if RECEIVEACK( ) then
            data.dequeue( )
            n_pkt ← n_pkt + 1
        end if
    end if
end function

function ONSLOTFRAMEEND( )                    ▷ Periodic cleanup
    a_slotframes ← max(0, a_slotframes − 1)
    if n_pkt = 0 OR a_slotframes = 0 then
        AP_s ← NULL                           ▷ Deselect the active AP
    end if
    n_pkt ← 0
end function
```

See Algorithms 1 and 2 for the definition of *Instant*. Its basic operation is also outlined in Fig. 3. In the pseudocode, SELECTAP( ) and SELECTWEARABLE( ) refer to the objective functions for selecting the access point and the wearable, respectively, and are implementation-defined. In our model, all access points have equal rank, *i.e.*, equal cost to deliver data to the sink. Under this assumption, given multiple access points that are willing to allocate slots for a wearable, from the wearable's point of view it is always best to select the AP with the best link. Conveniently, the capture effect ensures that the ACK that gets through in case of a collision always happens to be the one with the strongest signal. Hence, choosing and implementing SELECTAP( ) under our assumptions is straightforward. If the access points had different costs to the sink, this *cost-to-sink* could be added to the ACK packets, and used by the SELECTAP( ) function.

Selecting the wearable is not as straightforward; in this paper we consider only random selection, however, other methods and specific objective functions can be investigated

Algorithm 2 The Instant algorithm: access point operation

$W_s \leftarrow$ NULL ▷ The selected wearable
$a_{slotframes} \leftarrow 0$ ▷ Number of slotframes still selected
$n_{pkt} \leftarrow 0$ ▷ Number of packets Rx in last slotframe
$W_{active} \leftarrow \varnothing$ ▷ All active wearables
$p \leftarrow$ NULL ▷ Probing or data packet
$ack \leftarrow$ NULL ▷ Acknowledgment packet

function ONPROBINGTIMESLOT( ) ▷ Probing for localization
  $\langle p, rssi \rangle \leftarrow$ RECEIVEPACKET($O_{probing}$)
  if $p \neq$ NULL AND $p.queuesize > 0$ **then**
    ADDORUPDATEWEARABLE($W_{active}, p.address, rssi$)
    if $W_s =$ NULL **then**
      $\langle W_s, a_{slotframes} \rangle \leftarrow$SELECTWEARABLE($W_{active}$)
    end if
    $n_{subslot} \leftarrow$ RANDOMINRANGE$(0, N_a - 1)$
    $T_{delay} \leftarrow \tau_i + \tau_a \times n_{subslot}$
    DELAY(T_{delay})
    $ack \leftarrow$ CONSTRUCTACK( )
    if $W_s \neq$ NULL AND $Addr_{W_s} = p.address$ **then**
      $ack.a_{slotframes} \leftarrow a_{slotframes}$
    else
      $ack.a_{slotframes} \leftarrow 0$
    end if
    TRANSMITPACKET($ack, p.address, O_{probing}$)
  end if
end function

function ONUNICASTTIMESLOT( ) ▷ Data collection
  if $W_s \neq$ NULL **then**
    if RECEIVEPACKET(O_{W_s}) **then**
      $n_{pkt} \leftarrow n_{pkt} + 1$
    end if
  end if
end function

function ONSLOTFRAMEEND( ) ▷ Periodic cleanup
  $a_{slotframes} \leftarrow max(0, a_{slotframes} - 1)$
  if $n_{pkt} = 0$ OR $a_{slotframes} = 0$ **then**
    $W_s \leftarrow$ NULL ▷ Deselect the active wearable
  end if
  $n_{pkt} \leftarrow 0$
  $W_{active} \leftarrow$FILTER(W_{active}, T_{fresh})
end function

in the future: for example, selecting the wearable with the strongest RSSI, or with the most data to send.

An important tunable parameter of the algorithm is $a_{slotframes}$. This determines how fast an access point may switch between wearables. Small value means faster switching and makes the algorithm more flexible; large values are more suitable for networks with lower mobility. A value of infinity leads to *Instant* establishing permanent connections that are lost only when no further packets are received; however, the potential for fairness between wearables is lost.

Further in this paper we consider two modes of *Instant*: one where $a_{slotframes} = \infty$ (the *connection mode*) and one where $a_{slotframes}$ is in range $[1,5]$ – referred to as the *regular mode* or simply *Instant*. In the regular mode, the value of $a_{slotframes}$ is selected by each AP individually. It is equal to the number of slotframes since the last change in the AP's set of neighboring wearables, but bounded by 5 from above. To give an example, if the AP has just seen a new wearable, it will set $a_{slotframes} = 1$; if the neighborhood has remained the same for a long time, it will use $a_{slotframes} = 5$. This makes

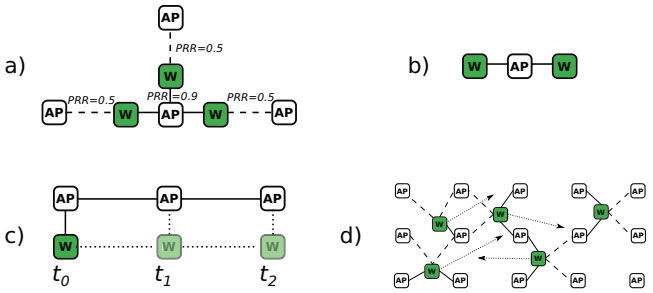


Figure 4. Several network topologies that benefit from *Instant*. (a) A topology where load balancing between the different AP is possible. (b) A neighborhood where the number of wearables is greater than the number of AP. (c) A mobile wearable passing between different AP. (b) A larger network with several mobile wearables.

the reaction speed of *Instant* proportional to the amount of mobility in the network. In contrast, the connection mode provides similar behavior to the BLE Connected Mode [7].

The proposed schedule leaves one slot in each slotframe reserved for non-*Instant* traffic (Fig. 3b). This can be used to exchange RPL and TSCH network maintenance packets between the AP, and provide coexistence between *Instant* and multihop routing. Reserving more than one slot for this is possible, but would proportionally reduce the data collection performance of *Instant*.

3.4 Discussion

Let us go back to the problem of collecting data from wearables. There are some common network topologies in which *Instant* shows its strengths for this application:

- Figure 4a. If RPL is used here, all wearables are likely to select the same access point as their routing parent and use it for data transfer (*e.g.*, all send to the central node, since they only have good links to that). *Instant* provides better throughput by utilizing more than one AP for data upload simultaneously. This happens both in the regular and in the connection mode. Furthermore, *Instant* could easily be extended to incorporate RSSI thresholding; it would allow the wearables to filter out access points with weak signals (*i.e.*, bad links, which lead to extra energy expenditure for retransmissions). By dynamically changing this parameter, the wearables would be able to select a tradeoff between energy-efficiency and system throughput.
- Figure 4b. Here *Instant* allows the two wearables to take turns when uploading data to the access point, in this way avoiding starvation of one of the wearables. In this way, *Instant* makes the network layer more suitable for responsive, interactive applications that run on wearables and communicate with the infrastructure in the real time.
- Figure 4c. *Instant* in the regular mode seamlessly switches between the different AP, so that at the different time moments t_0, t_1, and t_2 different AP are selected by the wearable. No explicit hand-off between AP is required. The connection mode adds just a small

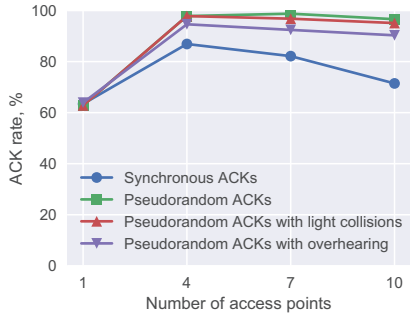
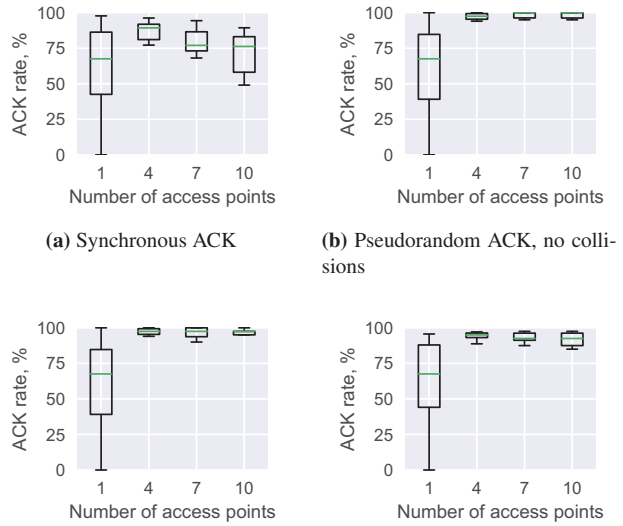


Figure 5. Comparison of the different anycast ACK techniques. Exp. results with single mobile wearable.



(a) Synchronous ACK

(b) Pseudorandom ACK, no collisions

(c) P-random ACK, light collisions

(d) Pseudorandom ACK, overhearing

Figure 6. Exp. results for the anycast ACK techniques depending on the number of access points. Each boxplot shows results from 7 rooms in the SPHERE House.

overhead, since the connection establishment and termination processes in *Instant* are light-weight. In each of the anycast probing slots information from multiple gateways is collected. Therefore, the wearables are able to discover new neighbors faster than if RPL was used.

- Figure 4d. RPL does not scale well to large networks due to the limitations in routing and neighbor table sizes on IoT devices [20], especially in the RPL storing mode. In contrast, *Instant* requires the nodes to keep neighbor information only for T_{fresh} slotframes, which is a small number (*e.g.*, 4 or 5 slotframes) in dynamic networks, thus reducing their memory requirements.

4 Single-Wearable Evaluation

We implement both *Instant* and the different anycast ACK techniques on top of Contiki-NG TSCH [12]. This section provides an experimental evaluation of these implementations, and the comparison between *Instant* and BLE.

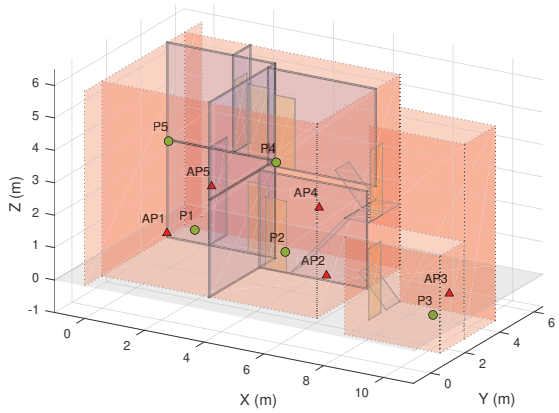


Figure 7. The positions of the wearable (Px) and the access points (APx) deployed in the SPHERE House testbed: living room (AP1), study (AP2), kitchen (AP3), guest bedroom (AP4), master bedroom (AP5).

4.1 Anycast ACK Techniques

In order to evaluate the reliability of anycast ACK, we perform an experiment in a test-bed deployed in a residential property (Fig. 7), named SPHERE House house for the remainder of this paper. We use a CC2650 evaluation module (EM) as the mobile wearable device, and compare the performance of the four different techniques (Fig. 2) depending on the number of AP in the house. All devices are transmitting with 0 dBm power. The experiment is designed so that all of the techniques are evaluated nearly simultaneously, thus avoiding the impact of medium- and long-term changes in the radio environment. In each active TSCH slot, there is a single ACK subslot for the synchronous approach. Odd-numbered slots also contain a subslot for each AP ("Pseudorandom ACK", no collisions); even numbered: $N_a = 4$ subslots divided between all of the AP. Slots divisible by 4 use overhearing ("Pseudorandom ACK with overhearing"), the other slots do not ("Pseudorandom ACK, light collisions").

Figure 5 shows the whole-house average results, Figure 6 the statistics about the different rooms. In the experiment with AP=1, only the AP2 in the center of the house is active; the other experiments iteratively add more active AP. Clearly, having just a single AP is not sufficient, as large parts of the house are out of its range. Due to the capture effect, the synchronous ACK technique is useful; however, its performance is worse than other options due to a larger number of collisions. The pseudorandom ACK (no collisions) technique performs the best; however, it is not practical, as with $N_a = 7$ or $N_a = 10$ there are too many ACK subslots; they cannot all be fitted in a standard-size 10 ms TSCH slot. In contrast, the approach with "light collisions" is both practical ($N_a = 4$) and shows almost as good performance. With this option, the number of colliding ACK transmissions remains small, allowing the capture effect to filter out the weaker ones. Finally, the overhearing approach reduces the ACK rate much more than the "light collisions" do – therefore the latter is preferable. In summary, we selected the "Pseudorandom ACK with light collisions" approach as the most practical one. We use this approach for the rest of the paper.

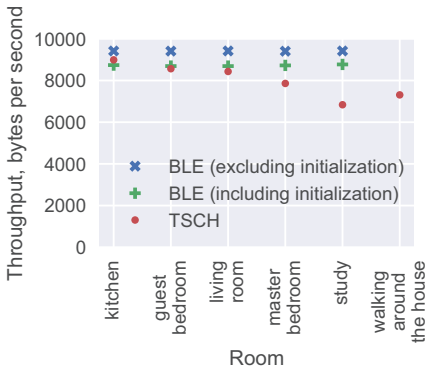


Figure 8. Application-level throughput in a single-wearable system. Exp. results using the setup in Fig. 7.

4.2 Throughput and Energy Efficiency

4.2.1 Experimental Setup

In this series of experiments, we compare the application-level throughput of TSCH (using *Instant*) with BLE. Both protocols use the same hardware: TI CC2650 EM as the wearable, custom TI CC2650 nodes as the access points. Transmit power is set to 0 dBm. The BLE setup uses the Simple Peripheral and Central example provided by Texas Instruments; the wearable is set to be the peripheral. Each connection interval (10 ms) the peripheral device attempts to send five notification packets, each of which contain 20 bytes of application data. We note that trying to further increase the throughput by sending more than five notification packets per connection interval caused the connection to fail after a couple of seconds.

In a single experiment, a single wearable is placed in the center of a room in the SPHERE House and sends application data as fast as possible to an access point located in the same room. The duration of an experiment is 2 min. It is repeated both for BLE and TSCH, and for each room in the house. In the final experiment, the wearable is moved around the house; this is done only for TSCH, as BLE does not support mobility between different AP out-of-the-box.

4.2.2 Throughput

Figure 8 shows the results. BLE requires approximately 200–400 milliseconds to establish a connection between the Central and Peripheral devices; the experiments take this into account and show results with and without this initialization delay. After a brief startup period, its datarate is very consistent: $\approx$ 9500 bytes per second, even though each room is different. TSCH results are with similar, slightly lower throughput, but much more variable.

The BLE results could be significantly improved by using extended packet sizes: the BLE 4.2 standard includes an optional LE data length extension that increases the maximal data payload size from 27 bytes to 251 bytes. However, this extension is optional and not supported by many BLE-capable devices, for example the BCM43438 BLE chip [2] used in Raspberry Pi Model 3B, therefore would adversely impact the interoperability of the solution.

Table 2. Application-level energy consumption. Experimental measurements on CC2650.

Protocol	Energy for 100 kB (STD)	Relative to TSCH
TSCH / Instant	164.87 mJ (0.19)	n/a
BLE Connected Mode	135.50 mJ (0.20)	−17.8 %

4.2.3 Energy Efficiency

Table 2 shows the energy required to collect 100 kB of data from the CC2650 device. The energy consumption is measured with a RocketLogger device [30] at 64 kS per second sampling rate. The measurements are repeated three times for each protocol; the average results and the standard deviations are given in the table. The energy per an application payload byte is 1.65 μJ for TSCH and 1.35 μJ for BLE. While BLE remains more efficient than TSCH, as expected due to its 4 times faster PHY datarate, the application-level difference is only 17.8 %, showing that the MAC layer overhead of TSCH is much smaller. The packet format of BLE is relatively inefficient, as larger proportion of the packet is taken up by headers, and the size of BLE ACK packets is much larger than the size of TSCH ACK packets, when measured proportionally to the packet payload size.

4.2.4 Discussion

The relatively small difference in energy consumption and throughput between TSCH and BLE in this single-stationary-wearable case (which arguably are the perfect conditions for BLE Connected Mode) allows to conclude that TSCH is a worthwhile challenger to BLE Connected Mode, in the application domain considered in this paper. Furthermore, there are qualitative reasons to prefer TSCH: robustness, flexibility, and openness. In more detail: first, the physical layer of the IEEE 802.15.4 standard is more robust than BLE due to using a more redundant modulation scheme. Second, as shown in the next section of this paper, for networks with multiple, mobile wearables, the connection-based communication model (implicitly enforced by the BLE Connected Mode) is too rigid and less efficient than *Instant* or other schemes possible on top of standard TSCH. Third, BLE requires that the implementers pay a licensing fee. This makes technological innovation on top of BLE harder compared to innovation on top of the fully open TSCH standards.

5 Multi-Wearable Evaluation

In this section, our implementation of *Instant* is compared with the RPL and Orchestra network stack. Cooja simulations are used to explore the performance in random scenarios, and the scaling properties of *Instant*; experiments with CC2650 nodes are used to validate these results in a test-bed.

5.1 Methodology

The main experimental settings are shown in the Table 3. For simulations, we use the *LogisticLoss* Cooja radio medium. This Cooja plugin uses the logistic function to model the PRR–RSSI relationship (as used previously by *e.g.*, Kim *et al.* [23]):

$$PRR(rssi) = \frac{1}{1 + exp(-(rssi - rssi_{50\,\%}))}, \qquad (2)$$

Table 3. Parameters used in the evaluation

	Parameter	Value
Main settings	Number of wearables	4
	Number of AP	5
	Data size	100 kB
	Packet payload	104 bytes
	HW platform	CC2650 EM
Instant	Initial $a_{slotframes}$	1
	Max $a_{slotframes}$	5
	N_a	3 subslots
	$\tau_{slotframe}$	50 slots
	T_{fresh}	4 slotframes
	Unicast slot proportion	90 %
Orchestra	Mode	Sender-based
	$\tau_{slotframe}$, broadcast	50 slots
	$\tau_{slotframe}$, unicast	50 slots
RPL	Implementation	"RPL Classic"
	Objective function	MRHOF
	Initial DIO period	2 sec
	Max DIO period	8 sec
	Probing interval	20 sec
	Max routes	16
TSCH	Number of channels	5
	Slot size	10000 μs
	Guard time	1800 μs
Simulations	Simulator	Cooja
	Platform	Cooja mote
	Radio Medium	Logistic Loss
	Wearable speed	1 m/s
	Max Tx range	20 m
	Capture effect	Yes
	Co-channel rejection	−3 dB
	Warm-up time	2 min
	Maximal duration	10 min

where *rssi* is the transmit power minus the path loss. To model the path loss $PL_{dBm}(d)$ we use the log-distance path loss model [28]:

$$PL_{dBm}(d) = PL_0 + 10 \cdot \alpha \cdot \log_{10} \frac{d}{d_0} + \mathcal{N}(0, \sigma), \qquad (3)$$

where the transmission range $d_0 = 20$ m, $PL_0 = -100$ dBm [1], $rssi_{50\%} = -92$ dBm, $\alpha = 3$, and $\sigma = 3$ [28].

One issue we face in the comparison is the large number of parameters of the Contiki-NG network stack, which are known to greatly impact network performance [31]. In order to make the comparison fair, some of the most important parameters of RPL (Table 3) are selected by evaluating them in the target simulations (Fig. 10, Fig. 11). To emulate the assumptions in *Instant*, we modify the MRHOF metric to make wearables consider all access points as having zero cost to the root. The Orchestra parameters, such as slotframe size, are set to match *Instant*.

Orchestra is not originally designed for high-throughput data collection; however, the burst mode from the IEEE 802.15.4 standard [4] makes it more suitable for this purpose. This burst mode is already enabled in the default Contiki-NG settings. However, the packet burst ends as soon as just one packet or one ACK is lost (Fig. 9a). To further optimize the performance of Orchestra for the target application, we add a simple tweak: a packet with the burst bit set always reserves the whole slotframe for the node. We call this modification the *Greedy Orchestra* (Fig. 9b).

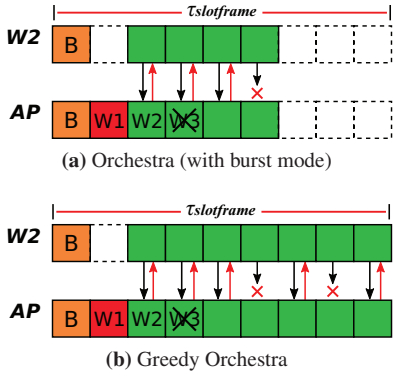


Figure 9. Our *Orchestra* schedule (a) and its greedy modification (b). B – broadcast slots; $W1$, $W2$, $W3$ – unicast slots initially allocated for specific wearables. Different colors denote different channels (c.f. Fig 3). In (a), a packet burst is terminated after the first loss of a packet or an ACK; in (b), the burst continues until the end of the slotframe.

5.2 Simulations

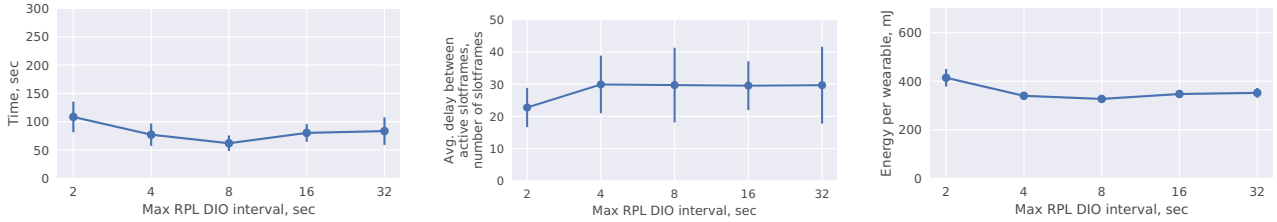
We compare the performance of the four different software options depending on the number of wearables, using the settings from Table 3. For each number of wearables, we use ten simulation files with random, but static wearable positions, and ten files where the wearables move according to the random waypoint mobility model. To simulate contention in the network, all wearables start trying to upload their data simultaneously, after a two-minute warm-up period, during which they join the TSCH network and, if RPL is used, establish routes. For results see Figures 12 and 13.

5.2.1 Data Collection Speed

We report the time it takes to collect 100 kB of data from each wearable. The timing starts at the moment when the wearables start trying to upload their data, and the experiment ends when the last wearable has uploaded all of its 100 kB. All data is collected in all experiments – the protocol does not lose any packets, *i.e.*, is 100 % reliable. In mobile scenarios, *Instant* both in the regular and in the connection mode shows data collection speed that is several times better than that of RPL+Greedy Orchestra. In static scenarios, the speed of *Instant* is similar to that of RPL+Greedy Orchestra, showing the additional mobility support does not adversely impact the ability to collect data from static node. As expected, the baseline RPL+Orchestra performs worse than the other options both in static and mobile scenarios.

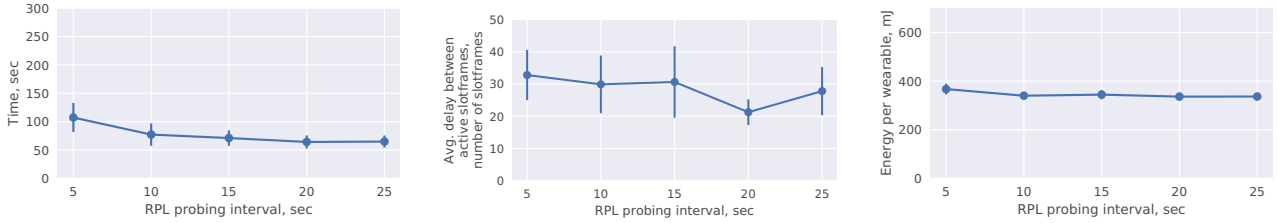
5.2.2 Fairness

For the purposes of this study, we are interested in fairness in the narrow sense of "avoiding starvation". In order to measure this metric, we look at the average duration of the time a wearable is without active unicast communication slots to any access point, *i.e.*, is suffering from starvation. For interactive applications that require communication between the



(a) Time to collect **(b)** Unfairness **(c)** Energy

Figure 10. RPL+Greedy Orchestra data collection performance depending on the RPL Max DIO interval. Here and further: the plots show mean values along with 90 % confidence intervals.



(a) Time to collect **(b)** Unfairness **(c)** Energy

Figure 11. RPL+Greedy Orchestra data collection performance depending on the RPL probing interval.

wearable and infrastructure, this metric is important for the responsiveness of the wearable from the user's point of view.

Instant in the regular mode demonstrates unequivocally better fairness than the other options in mobile scenarios, and similar results to those of the two RPL options in static networks. Due to the aggressive pairing between a single wearable and access point, the connection mode demonstrates worse fairness, especially when the number of wearables is larger than the number of access points.

5.2.3 Energy Consumption

To measure this metric, we only account for the CPU and radio energy consumption in active TSCH timeslots, using a bespoke energy model for TSCH on CC2650. The model is based on measurements with RocketLogger [30].

In the simulations, three of the four options show performance that is similar for all scenarios (*i.e.*, the mean value is within the same 90 % confidence range).

The connection mode fare worse, because in this mode, more packets are sent over bad links, therefore more retransmissions are needed. This happens because in this mode, the wearable does not break the link to its connected access point even when a better link becomes available.

5.2.4 Discussion

Overall, *Instant* shows the best performance in the mobile scenarios, and performance that is similar to RPL+Greedy Orchestra in the static ones. Despite accounting for the additional probing traffic, its energy consumption is not significantly worse. If the application needed to send out the periodic probing packets for other purposes (such as localization), the extra energy consumption would be negligible.

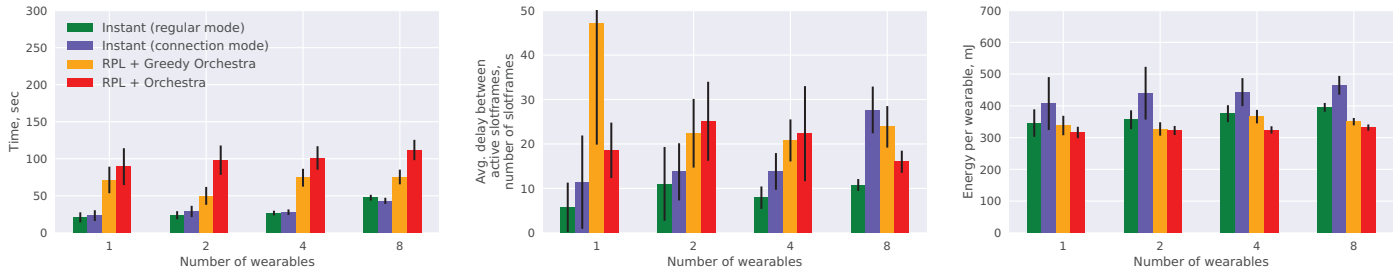
5.3 Experiments

We carry out the experiments in the SPHERE House. The static experiments take place in the kitchen (Fig. 18), the mobile – walking between the rooms in the house. The positions of the access points remain as in Figure 7. Given the relatively small size of the house, we only experiment with up to four wearables, and set the transmission power to −10 dBm to make the data collection more challenging. For each configuration, we repeat the 100 kB data collection four times. We do not evaluate RPL+Orchestra performance, since RPL+Greedy Orchestra showed much better speed in the simulations.

The results (Fig. 14, Fig. 15) show similar patterns as the simulation results, therefore validate the latter. There are two exceptions: much better average collection speed, and proportionally worse performance of RPL+Greedy Orchestra. The first is explained by the fact that the house is almost fully covered by the access points, therefore, unlike as in the simulations, there are no "pathological" situations when a wearable is outside of the communication ranges of all access points. The second is explained by high contention between the wearables when RPL+Greedy Orchestra is used. Most of the wearables select the same access point as their routing parent – *i.e.*, the one located in the center of the house (*AP*2 position in the Fig. 7). In contrast, *Instant* implicitly provides load balancing between all access points in a range. Similarly to the simulation results, no packets are lost (100 % delivery rate).

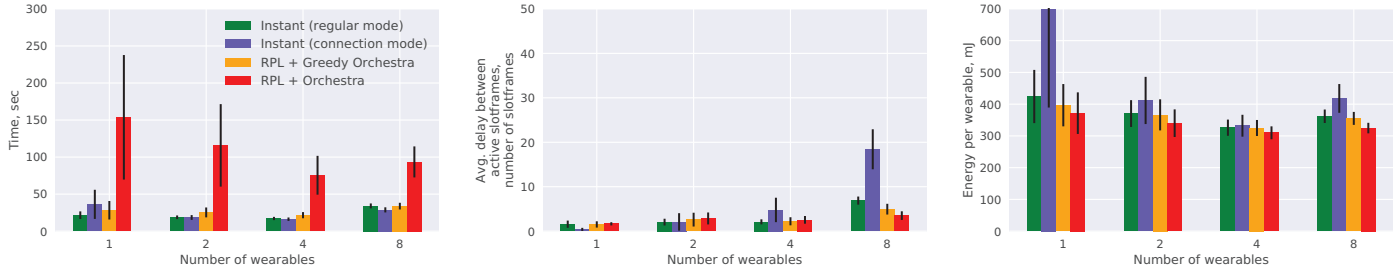
5.4 Scaling

We use further simulations to test the scaling properties of *Instant*. There are two sets of simulations, each of which changes the scale of the network in different dimensions. The first set of simulations changes the number of access points in the network along with the physical size of the network. The number of wearables is kept constant. The second set changes the density of the network in terms of wearables.



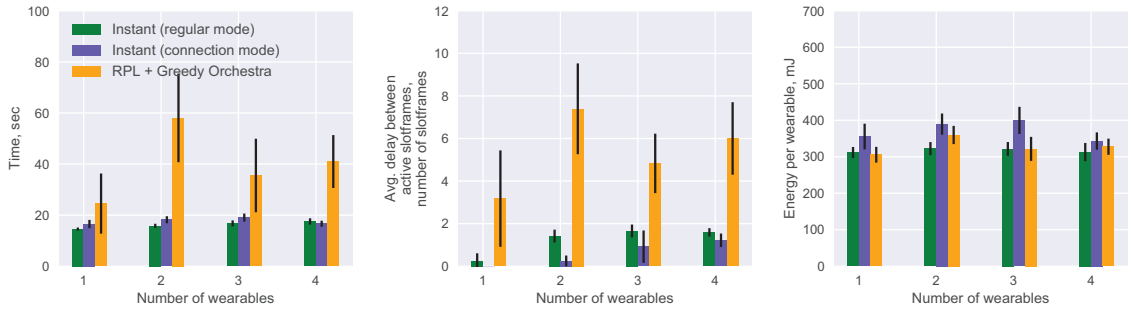
(a) Time to collect **(b)** Unfairness **(c)** Energy

Figure 12. Data collection performance. Results from simulations with mobile wearables.



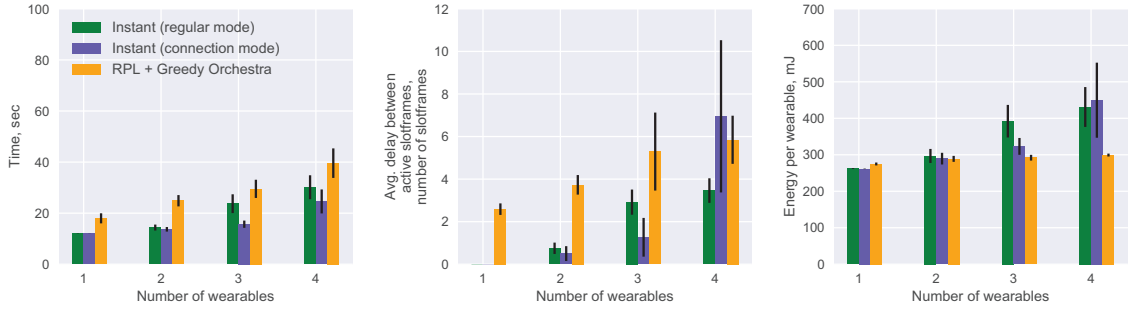
(a) Time to collect **(b)** Unfairness **(c)** Energy

Figure 13. Data collection performance. Results from simulations with static wearables.



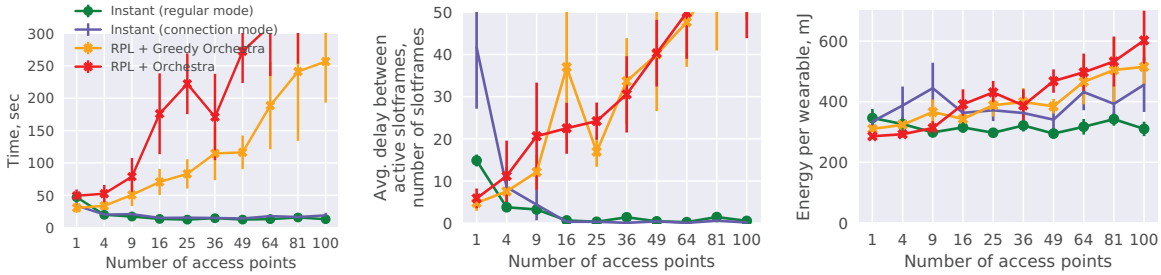
(a) Time to collect **(b)** Unfairness **(c)** Energy

Figure 14. Data collection perf. depending on the number of mobile wearables. Experimental results.



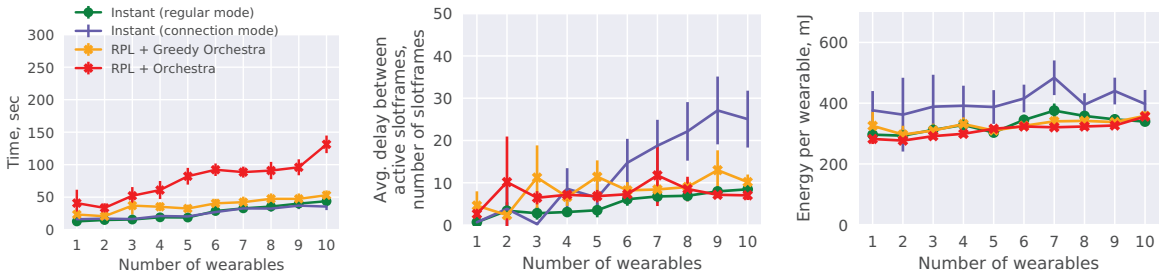
(a) Time to collect **(b)** Unfairness **(c)** Energy

Figure 15. Data collection perf. depending on the number of static wearables in a single room. Experimental results.



(a) Time to collect **(b)** Unfairness **(c)** Energy

Figure 16. Data collection performance depending on the number of access points. Results from simulations with mobile wearables. Access points arranged on a square grid with inter-device distance of 10 m. The results show similarly good performance with *Instant* in both modes as long as the number of access points is larger than the number of wearables.



(a) Time to collect **(b)** Unfairness **(c)** Energy

Figure 17. Data collection performance depending on the number of wearables. Simulations results, mobile wearables, with 4 access points arranged on square grid, inter-device distance 10 m. *Instant* in the regular mode generally shows better results than both variants of Orchestra; the connection mode shows high unfairness when there are >5 wearables.

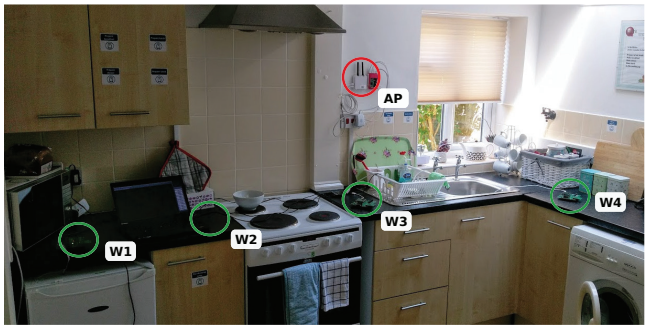


Figure 18. Kitchen area with four static wearables.

The number of access points is kept constant. In both experiments, we keep the other parameters to their default values, as given in the Table 3.

The results of the first set of simulations (Fig. 16) show that *Instant* is able to exploit the additional access points and increase its performance when the size of the network is increased. In contrast, the RPL options show much worse collection speed and fairness in larger networks. Surprisingly, increasing the routing table and neighbor table sizes (from 16 to 110 entries) in order to accommodate all potential neighbors decrease the performance in the RPL+Orchestra and RPL+Greedy Orchestra networks, so we report the results with 16 entries.

In contrast, *Instant* does not scale as well with the density of wearables (Fig. 17). When there are 6 or more wearables on 4 access points, it shows similar results as RPL+Greedy Orchestra. The connection mode in particular shows bad fairness when the number of wearables is larger than the number of access points. Improving the performance of *Instant* in denser networks is a future work.

6 Conclusion

In this paper we present *Instant*, a new distributed schedule for IEEE 802.15.4 TSCH. *Instant* shows consistently higher data collection speed from mobile nodes than existing options (RPL+Orchestra), even when the latter is specifically optimized for the target application. In particular, *Instant* achieves up to several times faster data collection even in small networks, and is able to further increase the speed in larger networks, as long as they also have larger numbers of access points. *Instant* similarly shows fairness and energy consumption that is either comparable with or better than those of RPL+Orchestra, both in mobile and in static networks. In some situations, *Instant* also achieves much better data collection speeds than RPL+Orchestra in static networks due to its implicit load balancing. Finally, the energy consumption and throughput of *Instant* are in the same class as those of BLE 4 Connected Mode, which is currently among the most popular options for data collection from mobile wearables. We hope that this work will lead to increasingly considering TSCH as a viable option for wearable applications.

7 Acknowledgments

This work was supported by the UK EPSRC Grant EP/K031910/1 and EurValve H2020 PHC-30-2015 689617, as well as ERDF Activity 1.1.1.2 “Post-doctoral Research Aid” (No. 1.1.1.2/VIAA/2/18/282). We thank Yang Bao, Antonis Vafeas, Ryan McConville, and Efstathios Mitskas for their help in the experiments.

8 References

[1] CC2650 SimpleLink Multistandard Wireless MCU. http://www.ti.com/lit/ds/symlink/cc2650.pdf.

[2] CYW43438 Single-Chip IEEE 802.11ac b/g/n MAC/Baseband/Radio with Integrated Bluetooth 4.1 and FM Receiver. http://www.cypress.com/file/298076/download.

[3] IPv6 over the TSCH mode of IEEE 802.15.4e IETF working group. https://tools.ietf.org/wg/6tisch/.

[4] IEEE Standard for Local and metropolitan area networks—Part 15.4. IEEE Std 802.15.42015, 2015.

[5] Y. Al-Nidawi and A. H. Kemp. Mobility aware framework for timeslotted channel hopping ieee 802.15. 4e sensor networks. *IEEE Sensors Journal*, 15(12):7112–7125, 2015.

[6] S. Baatz, M. Frank, R. Gopffarth, D. Kassatkine, P. Martini, M. Schetelig, and A. Vilavaara. Handoff support for mobility with ip over bluetooth. In *IEEE LCN*, pages 143–154, 2000.

[7] Bluetooth SIG. Specification of the Bluetooth System - Covered Core Package version: 4.0, 2010.

[8] D. Byrne, M. Kozlowski, R. Santos-Rodriguez, R. Piechocki, and I. Craddock. Residential wearable rssi and accelerometer measurements with detailed location annotations. *Scientific data*, 5:180168, 2018.

[9] T. Chang, M. Vucinic, X. Vilajosana, S. Duquennoy, and D. Dujovne. 6TiSCH Minimal Scheduling Function (MSF). Internet Draft, IETF, 2017.

[10] M. D’Souza, T. Wark, and M. Ros. Wireless localisation network for patient tracking. In *IEEE ISSNIP*, pages 79–84, 2008.

[11] S. Duquennoy, B. Al Nahas, O. Landsiedel, and T. Watteyne. Orchestra: Robust Mesh Networks Through Autonomously Scheduled TSCH. In *ACM SenSys*, pages 337–350. ACM, 2015.

[12] S. Duquennoy, A. Elsts, B. A. Nahas, and G. Oikonomou. TSCH and 6TiSCH for Contiki: Challenges, Design and Evaluation. In *13th Int. Conf. on Distributed Comput. in Sensor Syst. (DCOSS)*, 2017.

[13] A. Elsts, X. Fafoutis, P. Woznowski, E. Tonkin, G. Oikonomou, R. Piechocki, and I. Craddock. Enabling Healthcare in Smart Homes: The SPHERE IoT Network Infrastructure. *IEEE Communications Magazine*.

[14] X. Fafoutis, E. Tsimbalo, E. Mellios, G. Hilton, R. Piechocki, and I. Craddock. A residential maintenance-free long-term activity monitoring system for healthcare applications. *EURASIP Journal on Wireless Communications and Networking*, 2016(1):31, Jan 2016.

[15] F. Ferrari, M. Zimmerling, L. Mottola, and L. Thiele. Low-power wireless bus. In *ACM SenSys*, pages 1–14. ACM, 2012.

[16] C. Gezer, C. Buratti, and R. Verdone. Capture effect in IEEE 802.15.4 networks: Modelling and experimentation. In *Wireless Pervasive Computing (ISWPC), 2010 5th IEEE International Symposium on*, pages 204–209. IEEE, 2010.

[17] J. Haxhibeqiri, A. Karaagac, I. Moerman, and J. Hoebeke. Seamless roaming and guaranteed communication using a synchronized single-hop multi-gateway 802.15.4e tsch network. *Ad Hoc Networks*, 2018.

[18] S. R. Hussain, S. Mehnaz, S. Nirjon, and E. Bertino. Secure seamless bluetooth low energy connection migration for unmodified iot devices. *IEEE Transactions on Mobile Computing*, 17(4):927–944, April 2018.

[19] T. Huynh, F. Theoleyre, and W.-J. Hwang. On the interest of opportunistic anycast scheduling for wireless low power lossy networks. *Computer Communications*, 104:55–66, 2017.

[20] O. Iova, G. P. Picco, T. Istomin, and C. Kiraly. RPL, the Routing Standard for the Internet of Things... Or Is It? *IEEE Communications Magazine*, 17, 2016.

[21] P. Kakria, N. K. Tripathi, and P. Kitipawang. A real-time health monitoring system for remote cardiac patients using smartphone and wearable sensors. *Int. J. Telemedicine Appl.*, 2015:8:8–8:8, Jan. 2015.

[22] A. Kansal and U. B. Desai. Mobility support for bluetooth public access. In *2002 IEEE International Symposium on Circuits and Systems. Proceedings (Cat. No.02CH37353)*, volume 5, pages V–725–V–728 vol.5, 2002.

[23] S. M. Kim, S. Wang, and T. He. Exploiting causes and effects of wireless link correlation for better performance. In *IEEE INFOCOM*, pages 379–387, April 2015.

[24] T. J. M. Kooiman, M. L. Dontje, S. R. Sprenger, W. P. Krijnen, C. P. van der Schans, and M. de Groot. Reliability and validity of ten consumer activity trackers. *BMC Sports Science, Medicine and Rehabilitation*, 7(1):24, Oct 2015.

[25] P. Martin, B.-J. Ho, N. Grupen, S. Munoz, and M. Srivastava. An iBeacon primer for indoor localization: Demo abstract. In *ACM BuildSys*, pages 190–191. ACM, 2014.

[26] J. J. Oresko, Z. Jin, J. Cheng, S. Huang, Y. Sun, H. Duschl, and A. C. Cheng. A wearable smartphone-based platform for real-time cardiovascular disease detection via electrocardiogram processing. *IEEE Transactions on Information Technology in Biomedicine*, 14(3):734–740, May 2010.

[27] A. Pantelopoulos and N. G. Bourbakis. A survey on wearable sensor-based systems for health monitoring and prognosis. *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, 40(1):1–12, Jan 2010.

[28] J. S. Seybold. *Introduction to RF Propagation*. Wiley, 2005. ISBN: 0-471-65596-1.

[29] M. Siekkinen, M. Hiienkari, J. K. Nurminen, and J. Nieminen. How Low Energy is Bluetooth Low Energy? Comparative Measurements with ZigBee/802.15.4. In *Wireless Communications and Networking Conference Workshops (WCNCW), 2012 IEEE*, pages 232–237. IEEE, 2012.

[30] L. Sigrist, A. Gomez, R. Lim, S. Lippuner, M. Leubin, and L. Thiele. Measurement and validation of energy harvesting iot devices. In *Proceedings of the Conference on Design, Automation & Test in Europe*, pages 1159–1164. European Design and Automation Association, 2017.

[31] M. Strübe, F. Lukas, B. Li, and R. Kapitza. DrySim: simulation-aided deployment-specific tailoring of mote-class WSN software. In *ACM MSWiM*, pages 3–11, 2014.

[32] L. E. Talavera, M. Endler, I. Vasconcelos, R. Vasconcelos, M. Cunha, and F. J. d. S. e. Silva. The mobile hub concept: Enabling applications for the internet of mobile things. In *IEEE PerCom Workshops*, pages 123–128, March 2015.

[33] N. Twomey, T. Diethe, I. Craddock, and P. Flach. Unsupervised learning of sensor topologies for improving activity recognition in smart environments. *Neurocomputing*, 234:93–106, 2017.

[34] Q. Wang, X. Vilajosana, and T. Watteyne. 6TiSCH Operation Sublayer Protocol (6P). Internet Draft, IETF, 2018.

[35] T. Watteyne, L. Doherty, J. Simon, and K. Pister. Technical overview of SmartMesh IP. In *Innovative mobile and internet services in ubiquitous computing (IMIS), 2013 seventh international conference on*, pages 547–551. IEEE, 2013.

[36] C. S. Wong, I. Tan, R. D. Kumari, and F. Wey. Towards achieving fairness in the Linux scheduler. *ACM SIGOPS Operating Systems Review*, 42(5):34–43, 2008.

[37] D. Yuan and M. Hollick. Let’s talk together: Understanding concurrent transmission in wireless sensor networks. In *IEEE LCN*, pages 219–227. IEEE, 2013.

[38] Y.-L. Zheng, X.-R. Ding, C. C. Y. Poon, B. P. L. Lo, H. Zhang, X.-L. Zhou, G.-Z. Yang, N. Zhao, and Y.-T. Zhang. Unobtrusive sensing and wearable devices for health informatics. *IEEE Trans. on Biomedical Engineering*, 61(5):1538–1554, 2014.

[39] Y. Zhuang, J. Yang, Y. Li, L. Qi, and N. El-Sheimy. Smartphone-Based Indoor Localization with Bluetooth Low Energy Beacons. *Sensors*, 16(5), 2016.