Configuring Loosely Time-Triggered Wireless Control Software

Philipp H. Kindt
philipp.kindt@tum.de
TU Munich, Germany

Sumana Ghosh
sumana.ghosh@tum.de
TU Munich, Germany

Samarjit Chakraborty
samarjit@cs.unc.edu
University of North Carolina at Chapel Hill, USA

ABSTRACT

In many wireless control networks, sensor data and controller data are exchanged periodically, which requires periodic packet transmissions between the physical plant and the controller. As an alternative, event-triggered control paradigms imply that data is only exchanged when there are significant changes in the state of the plant, e.g., because of disturbances. This is the nature of many IoT scenarios and requires that a receiving device has to listen to the channel for incoming packets during all times. However, especially in mobile networks, in which all devices are battery-powered, continuous scanning would drain the battery quickly and hence, reception needs to be duty-cycled. When optimizing such duty-cycled operation, significant energy savings are possible using intelligent software-enabled communication scheduling. In this paper, we propose a wireless transmission scheme that supports *loosely time-triggered control*. When optimizing the scheduling of transmissions and reception windows in the communication protocol, our proposed scheme allows for energy-efficient communication without requiring strict clock-synchronization between the devices. We show that such a scheme is practical and can greatly reduce the energy consumption in event-triggered control applications.

ACM Reference Format:
Philipp H. Kindt, Sumana Ghosh, and Samarjit Chakraborty. 2020. Configuring Loosely Time-Triggered Wireless Control Software. In *23rd International Workshop on Software and Compilers for Embedded Systems (SCOPES '20), May 25–26, 2020, Sankt Goar, Germany*. ACM, New York, NY, USA, 4 pages. https://doi.org/10.1145/3378678.3391888

1 INTRODUCTION

Recently, many networked control systems use wireless connectivity for interconnecting sensors, controllers and actuators. Such networks are especially beneficial in Internet of Things (IoT) or smart home applications. For example, consider multiple sensors that are deployed within a building for sensing temperature, humidity and the open/close status of all windows. The sensed data is used to control heating using multiple battery-driven thermostats, such as the popular Google Nest [2]. Alternatively, consider a scenario in which the control algorithms are executed on a smartphone, thereby tailoring the control goals to the needs of different users. In all such scenarios, every participating device is energy-constrained. This imposes requirements on the wireless network, which we describe next.

Network Operation: Whereas a base-station without any energy constraints could always listen to the channel for incoming packets, energy-constrained devices need to apply a duty-cycled (on/off) reception pattern. Furthermore, the low penetration properties of protocols that operate in the 2.4 GHz band, e.g., WiFi, Bluetooth, Zigbee, etc., often induce the need for multi-hop communication. For realizing this, a device needs to relay all incoming data, which imposes the need for duty-cycled reception of mobile devices also when a base-station is present. When reception is duty-cycled, a device typically listens to the channel for a certain amount of time d every T_R time-units, as depicted in Figure 1. As a result, a packet can only be transmitted successfully within one of these reception windows. In the worst-case, a device has to wait for T_R time-units until its packet can be transmitted. This periodic communication scheme meets the requirements of synchronous, periodic control. In particular, when the period of the controller and T_R are identical and synchronized, the wireless network can achieve similar delays to those of a wired network. However, in *event-triggered* control strategies, data is only transmitted when significant changes in the state of the plant occur. For such paradigms, periodically transmitting packets is wasteful, since packets being sent in the absence of events carry no useful state or control information.

This Paper: The points in time at which packets are sent and the device listens to the channel are entirely defined by software. Hence, intelligent software that optimizes the communication schedule can significantly reduce the overall energy-consumption. In particular, current wireless protocols typically assume that a pair of devices are always synchronized with each other, i.e., every transmitted packet always meets a reception window of the opposite device. If now the transmitter is allowed to "skip" a certain number of instances of T_R, the maximum possible clock skew can become large. Hence, for event-triggered control, in which there are larger idle-phases without any packet transmissions, devices get out of synchronization. Existing strategies for nevertheless maintaining synchronization rely on extending the reception window [5]. This is energy-expensive and also limits the maximum allowed idle-time before synchronization is lost. As an alternative, the embedded software running on the nodes could drop the connection and re-synchronize when data is to be sent. In this paper, we propose a software-enabled networking paradigm that lies in between full synchronization and entirely dropping the connection. In particular, we propose a lightweight re-synchronization mechanism implemented in software, which exploits the fact that clock drift remains bounded even after longer idle periods. Hence, a re-synchronization mechanism that consumes less energy and induces a lower latency than a full re-synchronization can be carried out. The resulting protocol supports the paradigm of loosely time-triggered architectures (LTTA) [1] for mobile wireless control networks: every device can wirelessly dump its data at almost any point in time, while reducing the energy-consumption to an extent that makes LTTA feasible in mobile networks.

Paper Organization: The rest of this paper is organized as follows. In Section 2, we describe how current wireless protocol stack

SCOPES '20, May 25–26, 2020, Sankt Goar, Germany
2020. ACM ISBN 978-1-4503-7131-5/20/05…$15.00
https://doi.org/10.1145/3378678.3391888

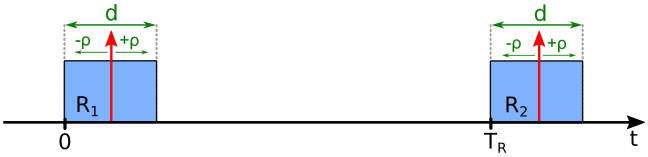

Figure 1: Duty-cycled communication. A device schedules its reception windows $R_1, R_2, \ldots$ with an interval T_R. Another device aims to transmit packets within the centers of the reception windows, which is aggrevated by a clock skew of $\rho \leq \rho_m$ between sender and receiver.

software addresses the clock synchronization problem. In Section 3, we describe our proposed solution. In Section 4, we present a case-study to show that our proposed solution is significantly more energy-efficient than known techniques. We conclude our findings in Section 5.

2 MAINTAINING SYNCHRONIZATION

Rather than periodically executing a control algorithm, it is often sufficient to compute new control input only when certain events occur. For example, a temperature sensor only needs to transmit its measured values to a controller when the difference from its previously transmitted value exceeds a certain threshold. When using a periodic reception pattern, a delay of up to T_R time-units is induced, since the transmission needs to be postponed until the next reception window arises. The value of T_R is determined by the maximally allowed delay and hence by stability and control performance requirements. In other words, if a certain bounded delay can be accepted from the control perspective, duty-cycled reception can be applied for event-triggered control.

While such a communication scheme is easy to realize on an ideal radio, real-world hardware is prone to clock inaccuracies. For being able to communicate, a transmitting device (T) needs to estimate the next point in time at which the receiving device (R) will schedule a reception window. This requires some form of clock synchronization among both devices. In particular, T's objective is to estimate the center of a reception window. Whenever R receives a packet from T, it can measure the relative clock skew ρ (cf. Figure 1) between both devices as the difference in time of expected and actual reception. Based on this, R can slightly adjust the beginning of its next reception window, thereby preventing an accumulation of the skew.

Typically, the clock accuracy on each device is specified by a maximum relative deviation Δ from its nominal value. The maximum skew ρ_m between two devices is given by

$$\rho_m(t) = 2 \cdot \Delta \cdot t, \tag{1}$$

where t is the time since the last packet from T has been received by R. In synchronous control systems with periodic transmissions, $t = T_R$ and hence, ρ_m always remains on a low level. In event triggered control, t can become considerably larger. For this reason, the following technique is commonly used.

2.1 Skipping Windows and Window Widening

Existing wireless protocols support that T may skip multiple instances of T_R without transmitting a packet, which is clearly beneficial for event-triggered control. For example, the Bluetooth Low Energy (BLE) protocol applies a concept called *slave latency* [5]. Here, T can skip up a certain number of reception windows in a row. To compensate for the clock skew, R estimates ρ_m based on the time since the last reception. Every reception window therefore starts ρ_m time-units earlier, while ending ρ_m time-units later. Hence, packets are still received also in the presence of clock skew. As a drawback, since every reception window is increased by $2 \cdot \rho_m(t)$, with t becoming larger in each instance of T_R, the energy consumption is significantly increased. Though becoming infeasible from an energy-perspective before, the idle-time is limited by the size of the reception window reaching T_R. This leads to a maximum possible idle-time t_m of $t_m = 1/4 \cdot (T_R - d)$.

2.2 Re-Synchronization

The natural alternative to dropping events is entirely giving up the synchronous connection and re-synchronizing opportunistically, once data is to be transmitted. The synchronization procedure relies on transmitting a sequence of probing packets using a pattern that guarantees an overlap with a reception window within bounded time. It has recently been shown in [3] that the minimum number M of packets that need to be transmitted for guaranteeing that a probing packet overlaps with a reception window is as follows.

$$M = \left\lceil \frac{T_R}{d} \right\rceil \tag{2}$$

Practically, we can send M synchronization packets within T_R time-units to guarantee re-synchronization without violating our latency constraints. We will discuss the transmission pattern of these M packets in the next section. The receiving device R uses the same values of T_R or d as for exchanging packets synchronously and hence, no additional energy is required for R. Clearly, the energy needed for synchronization required by T is given by the number of packets M and hence does not depend on the amount of clock skew. If the idle-time t becomes sufficiently large, a re-synchronization might consume less energy than maintaining a synchronous connection using window-widening.

Strategies that, depending on the current value of t, drop the synchronization and re-synchronize if needed, have not yet received significant attention. It is worth mentioning that different values of T_R and d, which might deviate from what is used in a synchronous connection, lead to a different energy-consumption of T and R. However, solutions with reduced energy consumption also lead to higher latencies, and we require a maximum latency of T_R, as in a synchronous connection. In many cases, a full re-synchronization is not necessary, since some degree of synchronization has remained since the last encounter. We next describe a solution exploiting this insight, which further reduces the energy consumption.

3 LOOSELY-SYNCHRONOUS COMMUNICATION

Equation 2 can be derived as follows. The first packet sent in a sequence of packets for synchronization has a random time-offset Φ between 0 and T_R from the closest scan window that is temporally on its left (cf. Figure 2). Whenever such a packet is sent without being received successfully, we can exclude a contiguous range of initial offset of length d, i.e., no beacons sent with such an offset can overlap with the reception window (see [3] for details). Hence,

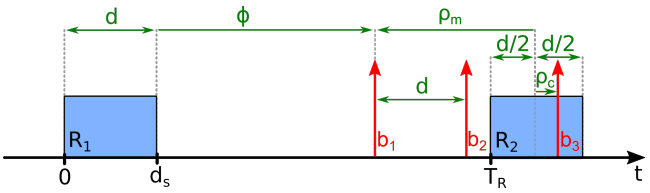

Figure 2: Partial Re-Synchronization

for probing all possible offsets, $M = \lceil \frac{T_R}{d} \rceil$ packets are needed. If the maximum possible shift between the center of the reception window and the first beacon in a synchronization sequence is bounded to $\pm\rho_m$, also the range of possible offsets Φ is constrained. As a result, the number of beacons to be sent can be reduced, as described next.

3.1 Adaptive Re-Synchronization

Let us assume that T sends a packet after an idle-time of t time-units, which leads to a maximum clock drift of $\pm\rho_m$, as given by Equation 1. We know that the packet that is supposed to be sent at the center of d may be sent by ρ_m time-units too early (negative skew) or too late (positive skew) due to the clock drift. We can exploit this insight to schedule our packets for synchronization. Our proposed scheme for achieving re-synchronization with $M' \leq M$ packets, which is depicted in Figure 2, works as follows.

(1) Assume an idle-time of t, after which T attempts to transmit a packet to R.
(2) Before transmitting, T estimates ρ_m (based on the idle-time t and using Equation 1) and artificially sends its packet by ρ_m time-units too early.
(3) In the best case, there is a positive clock skew of ρ_m time-units (i.e., a delay) and hence, this packet overlaps with d.
(4) To account for all other potential values of the clock skew, we re-transmit the packet M' times. The distance between two consecutive packets is given by d. When accounting for the worst-case, where the clock skew has a value of $-\rho_m$, the number M' of synchronization packets is as follows.

$$M' = \left\lceil \frac{2 \cdot \rho_m - d/2}{d} \right\rceil + 1 \tag{3}$$

(5) For "resetting" the idle time t for the next transmission, every packet piggy-backs a counter value k, which represents the packet count in the current sequence of synchronization packets. R will receive a packet that contains a certain counter value k, using which it can compute the actual clock skew ρ that has occurred as

$$\rho = -\rho_m + k \cdot d + \rho_c, \tag{4}$$

where ρ_c is the difference of the reception time of the received packet from the center of d (cf. Figure 2).

In the above, we have implicitly assumed that a packet being sent such that it protrudes beyond d is also received successfully, since d will be extended upon a partial reception.

Clearly, $M' \leq M$, as long as $\rho_m \leq (\lceil T_R/d \rceil - 1/2) \cdot d$. Therefore, whenever $\rho_m > (\lceil T_R/d \rceil - 1/2)$, a full re-synchronization is applied by sending M consecutive beacons with a spacing of d time-units, while the scheme described above is applied for smaller values of ρ_m. Both techniques bound the delay to T_R time-units.

With the described scheme, T can send control data with a granularity of T_R time-units, while being allowed to skip an arbitrary number of reception windows. We call this scheme *adaptive re-synchronization*. It supports *loosely time-triggered* communication, since the communication is carried out in a loosely synchronized manner using the periodic reception windows, while an event-triggered control system can initiate a transmission with a sufficiently fine time granularity. Note that the described adaptive re-synchronization scheme provides *disjoint coverage* (i.e., for no clock skew, no more than one beacon is received successfully). Hence, the re-synchronization procedure provides the lowest possible energy consumption for the given latency (see [4] for details). In the next section, we compare the energy efficiency of the overall scheme to the alternative ones.

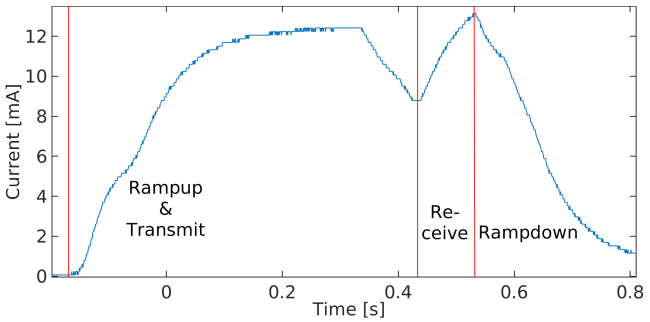

Figure 3: BLE advertising packet

4 PERFORMANCE EVALUATION

In this section, we evaluate the energy consumption of the 3 described strategies, viz., (i) not maintaining any synchronization an re-synchronizing on demand, (ii) maintaining a synchronous connection using window-widening, and (iii) adaptive re-synchronization. We first discuss the energy characteristics of wireless radios.

4.1 Energy Model

When a radio listens for incoming packets in a synchronous connection, there is a minimal idle-listening duration, below which an incoming packet is risked to be overheard. We can estimate this length by accounting for three components. First, we assume that a packet is detected upon the reception of a 1-byte preamble, as specified for BLE [5]. Second, we assume that our radio uses a common 32 768 Hz clock for scheduling reception windows. We therefore assume that our minimal reception window has an additional length of two clock ticks to compensate for quantization errors. Third, we compute and add the window widening needed to compensate for the clock-skew that arises within $1 \cdot T_R$. We assume a clock accuracy of $\Delta = 500 \cdot 10^{-6}$, which is equal to the minimum requirement for BLE. On a 1 MBit/s radio, we obtain a reception window for idle-listening of length $d = 169\,\mu s$, which is extended by the time to receive the additional bytes upon a partial reception.

Figure 3 depicts the measured current of a Nordic nRF518222 BLE radio. There are two distinguishable peaks. The first one is related to transmitting a BLE packet, the second one is related to listening for a potential response. As can be seen, in addition to the energy needed for actually transmitting and receiving packets, wireless radios induce various overheads. The overheads required for running the BLE protocol on a Bluegiga BLE112 radio have been measured in [4]. We reuse these overheads for estimating the energy consumption in our considered scenario (particularly, we account for the *head*,

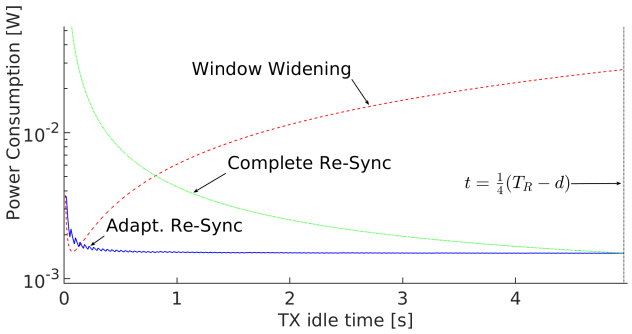

Figure 4: Joint power consumption for reception and transmission for different strategies with T_R = 10 ms.

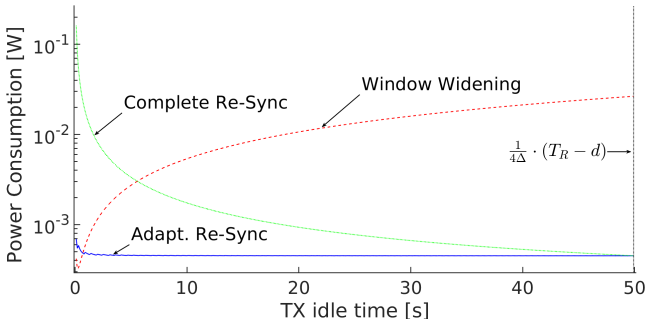

Figure 5: Joint power consumption for reception and transmission for different strategies with T_R = 100 ms.

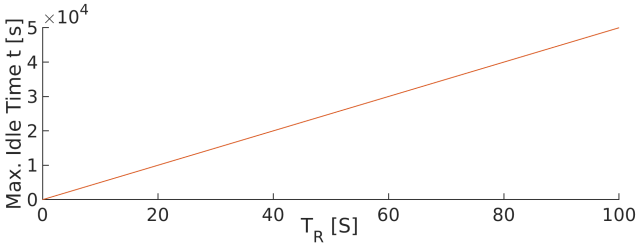

Figure 6: Max. idle time until which adaptive re-synchronization is beneficial.

cpre, *prerx*, *tra*, *tail*, *to* overheads upon reception and *head*, *cpre*, *pretx*, *tra*, *tail*, *to* overheads upon transmission, as described in [4]). For example, according to [4], idle-listening takes an effective duration of 123 µs, in which the radio consumes its characteristic reception current. Note that this duration is not identical to the actual length of d, since it accounts for various overheads. Though the values from [4] might differ slightly for non-BLE operation, they will allow for a reasonable comparison of our strategies. We further assume that each transmitted packet contains 10 bytes, which already include the 1-byte preamble. Further, we assume an operating voltage of 3 V.

Using this, we can estimate the power consumed by the different strategies. For the **window-widening strategy**, the power demand for transmission is given by $1/t \cdot Q_{pkg} \cdot 3\,\text{V}$, were Q_{pkg} is the charge consumed for transmitting one packet, including all overheads. For reception, the window-widening strategy incurs a power of $1/T_R \cdot (\rho_m \cdot 2 \cdot I_{RX} + Q_{idle}) + 1/t \cdot 9\,\text{byte} \cdot 8\,\mu\text{s/byte}$, with I_{RX} being the current consumption for reception and Q_{idle} the charge spent for idle-listening. For the **complete re-synchronization strategy**, the power spent for transmission is given by $1/t \cdot M \cdot Q_{pkg} \cdot 3\,\text{V}$. For reception, the power is given by $1/T_R \cdot (2\rho_m \cdot I_{RX} + Q_{idle}) + 1/t \cdot 9\,\text{byte} \cdot 8\,\mu\text{s/byte}$. The **adaptive re-synchronization strategy** dissipates the same power for reception, while the power for transmission is given by $1/t \cdot M' \cdot Q_{pkg} \cdot 3\,\text{V}$. The constant of 9 byte in the equations above represents the packet length of 10 byte, from which 1 byte has to be subtracted, since Q_{idle} already accounts for listening for the 1-byte preamble.

4.2 Evaluation

We first consider the joint power demand for transmission and reception for a maximum latency of T_R = 10 ms. Figure 4 depicts the power demand of the different strategies. As can be seen, the window widening strategy is only beneficial for very short idle-times t below around 150 ms. For all other idle times, the adaptive re-synchronization strategy significantly outperforms the other strategies. For $t \longrightarrow 1/4\Delta(T_r - d)$, a complete re-synchronization after every t time-units incurs a similar overall power demand as the adaptive re-synchronization strategy.

Figure 5 depicts the results for a larger maximum latency of T_R = 100 ms. As can be seen, the results look similar, while the idle-time until which adaptive re-synchronization reduces the power demand compared to a complete re-synchronization every t time-units can be increased significantly. Figure 6 evaluates the maximum idle time until which energy savings can be achieved using the adaptive re-synchronization strategy, compared to a full re-synchronization. We study different values of T_R. Especially in applications with low latency demands, energy can be saved even for very large idle-times. For example, if a latency of 20 s can be tolerated, idle-times of up to 2.8 h can be realized.

5 CONCLUDING REMARKS

Our results show that adaptive re-synchronization can be very beneficial for event-triggered control applications with slow dynamics. If latencies of a few seconds can be tolerated, this technique allows for low-energy operation with hours of idle-time. Even in scenarios with faster dynamics and maximum tolerable latencies of around 10 ms, our scheme can save energy when packets are exchanged with an average period of up to multiple seconds. Therefore, this scheme is especially beneficial in IoT, smart-home applications, e.g., controlling heaters using temperature sensors, and industrial control systems. Further research needs to measure these energy savings on real-world hardware for obtaining more accurate results.

REFERENCES

[1] A. Benveniste, P. Caspi, M. Di Natale, C. Pinello, A. Sangiovanni-Vincentelli, and S. Tripakis. 2007. Loosely time-triggered architectures based on communication-by-sampling. In *IEEE International Conference on Embedded Software (EMSOFT)*. 231–239.

[2] Google, Inc. 2020. Google Nest Learning Thermostat. (2020). store.google.com/us/product/nest_learning_thermostat_3rd_gen_specs.

[3] P. H. Kindt and S. Chakraborty. 2019. On Optimal Neighbor Discovery. In *ACM Special Interest Group on Data Communication (SIGCOMM)*.

[4] P. H. Kindt, D. Yunge, R. Diemer, and S. Chakraborty. 2020. Energy Modeling for the Bluetooth Low Energy Protocol. *ACM Trans. Embed. Comput. Syst. (TECS)* 19, 2 (March 2020).

[5] Bluetooth SIG. 2019. Specification of the Bluetooth System 5.2. (December 2019). Volume 0, available via bluetooth.org.