TTW: A Time-Triggered Wireless Design for CPS

Romain Jacob∗ Licong Zhang† Marco Zimmerling† Jan Beutel∗ Samarjit Chakraborty† Lothar Thiele∗

∗ETH Zurich, Switzerland firstname.lastname@tik.ee.ethz.ch
†TU Munich, Germany firstname.lastname@rcs.ei.tum.de
‡TU Dresden, Germany marco.zimmerling@tu-dresden.de

Abstract—Wired fieldbuses have long been proven effective in supporting Cyber-Physical Systems (CPS). However, various domains are now striving for wireless solutions due to ease of deployment or novel functionality requiring the ability to support mobile devices. Low-power wireless protocols have been proposed in response to this need, but requirements of a large class of CPS applications can still not be satisfied. We thus propose Time-Triggered Wireless (TTW), a distributed low-power wireless system design that minimizes communication energy consumption and offers end-to-end timing predictability, runtime adaptability, reliability, and low latency. Evaluation shows a 2× reduction in communication latency and 33-40% lower radio-on time compared with DRP, the closest related work, validating the suitability of TTW for new exciting wireless CPS applications.

I. INTRODUCTION

Many application domains, in which wireline communication systems like fieldbuses have been the norm for several decades, are now striving for wireless solutions due to their flexibility, cost efficiency, and low installation and maintenance efforts [1]. Indeed, international dependability competitions have revealed excellent performance, reliability, and energy efficiency of state-of-the-art wireless solutions despite heavy interference [2]. Moreover, there exist wireless networking substrates providing runtime adaptability and end-to-end communication guarantees [3]–[5] previously thought difficult or even impossible [6].

Unfortunately, the minimum latencies achievable by existing wireless solutions between distributed application tasks are often too long to support, for example, the update rates required for closed-loop control [1], [7] and swarm coordination [8]. Is it possible to cut latency without abandoning adaptability?

Wireless nodes typically duty-cycle their radio transceivers to save energy. Thus, neighboring nodes require overlapping wake-up intervals to communicate. This motivates a common wireless design that minimizes energy consumption by using rounds, i.e., time intervals where all nodes wake up, exchange messages, and then turn off their radios [9]–[11]. A scheduling policy defines when the rounds take place and which nodes are allowed to send messages in each round. However, CPS also execute tasks, e.g., for sensing or actuation, and the application requirements are often specified end-to-end between these distributed tasks. Meeting such requirements calls for co-scheduling the execution of tasks and the transmission of messages as previously proposed for wired systems [12]–[14].

We also consider a round-based design. Three challenges arise. First, state-of-the-art scheduling methods for wired buses are not directly applicable. They assume that communication can be scheduled at any point in time, which does not conform to the concept of pre-scheduled rounds in a wireless setting. Second, the optimization problem for co-scheduling tasks and messages cannot be solved online in a low-power setting. Third, pre-computed schedules are often desirable in hard real-time systems, yet CPS often require runtime adaptability while remaining predictable even in case of packet losses.

We propose Time-Triggered Wireless (TTW), a novel low-power wireless system design that solves these challenges and meets the requirements of CPS. This paper presents the main concepts behind TTW, including its foundations and additional techniques it uses to achieve low end-to-end latencies, runtime adaptability, energy efficiency, and reliability (Sec. II). Sec. III describes how we adapt the co-scheduling methods invented for wired systems to round-based wireless systems. Using fine-grained time and energy models, we show in Sec. IV a 33-40% reduction in radio-on time due to rounds, and 2× shorter minimum communication latencies than the state of the art. We review related work in Sec. V then conclude.

II. SYSTEM DESIGN

This section lays the foundations for TTW and presents the additional concepts it uses to achieve low end-to-end latency, runtime adaptability, energy efficiency, and reliability.

TTW Foundations. Let us consider a set of nodes connected by a wireless multi-hop network as shown in Fig. 1(a). Distributed applications that are composed of multiple tasks execute on the network. Tasks are mapped to nodes and exchange messages wirelessly. To minimize the energy consumed for wireless communication, we group message transmissions into communication rounds, i.e., time intervals where all nodes turn on their radios and communicate. Rounds are composed of (up to) $B$ contention-free slots. Within each slot, the communication of a single message is realized by one network-wide Glossy flood [15] as illustrated in Fig. 1(b). The network is controlled centrally by a node called the host. Commands are sent by the host at the beginning of each round in an additional slot called beacon that carries the schedule information for that round.

TTW inherits these concepts from the Low-power Wireless Bus (LWB) [10], which has several benefits. It is based on Glossy, which is highly reliable, energy efficient [2] and provides sub-microsecond time synchronization accuracy [15]. As flooding a packet using Glossy is independent of the network state, it creates a virtual single-hop network, where every node can directly communicate with every other node. This allows to schedule LWB like a shared bus. Finally, as each message is received by all nodes, LWB seamlessly supports unicast, multicast and broadcast transmissions. For a given packet size, the transmission time only depends on the network diameter.

Protocols built on top of LWB can provide communication guarantees between network interfaces [3], [4] and application...
Building-up TTW. To achieve low end-to-end latency among distributed applications tasks (e.g., 10-500 ms for a distributed closed-loop control system [7]) TTW co-schedules task executions and message transmissions together with the communication rounds. This scheduling problem is a complex optimization that cannot be solved on-line, even less in a low-power setting. Therefore, TTW statically synthesizes the schedule of all tasks, messages, and rounds in order to meet the application’s real-time constraints, to minimize communication energy costs, and to minimize end-to-end latency. Moreover, TTW satisfies the runtime adaptability requirement of CPS applications by switching between multiple pre-configured operation modes, similar to [16]. See an extended report for more details [17].

Offline scheduling allows TTW to minimize the communication overhead and hence energy consumption by distributing all schedules at deployment time rather than at runtime. This way, TTW also improves reliability: Since the schedules are pre-distributed, it is sufficient for any node to receive one short beacon to retrieve the system state (i.e., the schedule phase) and thus to know which message is to be sent in which slot and when to wake up for the next round. If a node does not receive a beacon, it does not participate in the communication.

Altogether, these concepts guarantee that TTW executes predictably and reliably. In contrast to LWB [10], TTW guarantees safe operation in terms of non-overlapping communication (i.e., no contention slot [10]), combines task and communication scheduling, and supports safe and fast mode switches.

III. SYSTEM MODEL AND SCHEDULING PROBLEM

TTW statically synthesizes the schedule of all tasks, messages, and communication rounds (Sec. II) using Integer Linear Programming (ILP) methods inspired by the wired domain and adapted to support communication rounds. This section formalizes TTW’s system model, then describes the challenge of combining ILP with rounds and shows how we address it. The complete ILP formulation can be found in [17].

System Model. Each distributed application is composed of tasks and messages connected by precedence constraints described by a directed acyclic graph, where vertices and edges represent tasks and messages, respectively. We denote by \( a.G \) the precedence graph of application \( a \). Each application executes at a periodic interval \( a.p \), called the period. An application execution is completed when all tasks in \( a.G \) have been executed. All tasks and messages in \( a.G \) share the same period \( a.p \). Applications are subject to real-time constraints. The application’s relative deadline, denoted by \( a.d \) with \( a.d \leq a.p \), represents the maximum tolerable end-to-end latency to complete the execution.

A node always executes at most one non-preemptive task. Each task \( \tau \) is mapped to a given node \( \tau.map \) on which it executes within a worst-case execution time (WCET) \( \tau.e \). The task offset \( \tau.o \) represents the start of the task execution relative to the beginning of the application execution. A task \( \tau \) can have an arbitrary number of preceding messages, i.e., messages that must be received before \( \tau \) can start executing (Fig. 2).

Every message \( m \) has at least one preceding task. The message offset \( m.o \) relative to the beginning of the application execution represents the earliest time the message \( m \) can be allocated to a round. The message deadline \( m.d \) relative to the message offset represents the latest time when the message transmission must be completed. All messages have the same payload. Within one application \( a \), each task is unique but messages may not be: Multiple edges of \( a.G \) can be labeled with the same message \( m \), which captures the case of multicast or broadcast communication (e.g., \( m_2 \) in Fig. 2).

Operation modes represent mutually exclusive phases of the system, e.g., normal, and emergency modes, each executing a specific schedule. A mode \( M \) is a set of applications that are concurrently executed. The mode’s hyperperiod is the least common multiple of the mode’s applications.

Finally, the schedule of mode \( M \) contains \( R_M \) communication rounds \( r \). Rounds are atomic (i.e., they cannot be interrupted) and composed of (up to) \( B \) slots, each allocated to a unique message \( m \). This results in a maximum round length \( T_r \). The starting time of round \( r \), denoted by \( r.t \), is the start of the round relative to the beginning of the mode’s hyperperiod. The allocation vector \( r.[B] \) is a vector of size \( B \) containing the ids of the messages allocated to the slots in round \( r \).

All modes, applications, task mappings, and WCETs are inputs of the problem. For a given mode \( M \), the remaining variables define the mode schedule, denoted by \( Sched(M) \):

\[
Sched(M) = \{ \begin{align*}
&\tau.o, m.o, m.d \\
&\tau.k.t, \tau.k.[B] & a \in M, (\tau, m) \in a.G \\
&k \in [1, R_M]
\end{align*}
\]

The scheduling problem entails finding an optimized mode schedule \( Sched(M) \) where

(i) application executions always meet their deadline, and
(ii) the number \( R_M \) of communication rounds is minimized.

Scheduling TTW. The schedule of a mode \( M \) is computed for one hyperperiod, after which it repeats itself. To minimize the number of rounds used, we solve the problem sequentially.
Each ILP formulation uses a fixed number of rounds $R_M$ per hyperperiod, starting with $R_M = 0$. The number of rounds is incremented until a feasible solution is found, or until the maximum number of rounds $R_{max}$ possible is reached. Thus, this approach guarantees by construction that if the problem is feasible, then the synthesized schedule is optimal in terms of number of rounds used. End-to-end latency is then minimized by setting the sum of application latencies as objective.

The ILP formulation includes several scheduling constraints that can be easily expressed in our system model: precedence constraints, end-to-end deadlines, no overlapping rounds/tasks, no more than $B$ messages per round. However, we must also guarantee that the allocation of messages to rounds is valid, that is, every message must be served in a round that starts after its release time (C1) and finishes before its deadline (C2).

These two constraints lead to a non-linear coupling between the variables and makes the ILP formulation non-trivial.

To address this challenge, we use the network calculus [18] concepts of arrival, demand, and service functions: $af$, $df$, and $sf$. Those functions count the number of message instances released, served, and with passed deadlines since the beginning of the hyperperiod, respectively (Fig.3). One can reformulate (C1) and (C2) with inequalities on $af$, $df$, and $sf$ (see [17] for details). However, these are step functions; they cannot be used directly in an ILP formulation. We work around this issue by adding variables to the formulation, which we constrain to be equal to the values of $af$, $df$, and $sf$ at relevant time points, i.e., when rounds start and end. These additional variables then enable the expression of (C1) and (C2) as linear constraints, which resolves the ILP formulation problem.

IV. Evaluation

We evaluate the performance of TTW regarding two criteria: the minimal achievable latency for an application, and the energy savings from the use of communication rounds.

Let $a.\delta$ denote the latency of an application $a$. Let $a.c$ be a chain in $a.G$. A chain is a path of $a.G$ starting and ending with a task without predecessor and successor, respectively. For example, $\tau_2, m_2, \tau_3$ is a chain of $G$ in Fig.2. In TTW the minimum achievable latency for one message is $T_r$. Thus then end-to-end latency $a.\delta$ is lower-bounded as follows:

$$ a.\delta \geq \max_{a.c \in a.G} \left( \sum_{\tau \in a.c} \tau.e + \sum_{m \in a.c} T_m \right) $$

To our knowledge, [5] is the only other wireless approach that provides end-to-end timing guarantees between application tasks. However, the best possible guarantee for a message is of the order of $2 \times T_r$ [5] due to the loose coupling between the task and message schedules. Hence, our approach improves the prior guarantee on message latency by at least a factor $2 \times$.

To minimize the application latency, one can minimize the round length $T_r$ down to a limit that we investigate now. A round is composed of one beacon and $B$ slots (Fig.1) of length $T_{slot}(l)$, where $l$ is the payload size in Bytes. The composition of each slot is detailed in Fig.4. First, all nodes wake up, which takes $T_{wake-up}$, and switch on their radio, which takes $T_{start}$. Then the flooding protocol starts (Fig.1(b)). We denote by $T_{hop}$ the time it takes for the flood to propagate one hop deeper into the network. The total length of the flood is

$$ T_{flood} = (H + 2N - 1) \times T_{hop} $$

with $H$ the network diameter and $N$ the number of times every node transmits each packet. $T_{hop}$ is itself divided into

$$ T_{hop} = T_d + T_{cal} + T_{header} + T_{payload} $$

where $T_d$ is a radio delay, and $T_{cal}$, $T_{header}$, and $T_{payload}$ are the transmission times of the clock calibration message, the protocol header, and the message payload, respectively. With a bit rate of $R_{bit}$, the transmission of $l$ Bytes takes

$$ T(l) = 8l / R_{bit} $$

Once the flood is completed, some time $T_{gap}$ is necessary to process the received packet. We split $T_{slot}$ into $T^{on}$ and $T^{off}$, the times spent with radio on and off, respectively. Thus, $T_{slot}(l) = T^{off} + T^{on}(l)$ with

$$ T^{off} = T_{wake-up} + T_{gap} $$

and

$$ T^{on}(l) = T_{start} + (H + 2N - 1) \times (T_d + 8L_{cal} + L_{header}s + l) / R_{bit} $$

finally

$$ T_r(l) = T_{slot}(L_{beacon}) + B \times T_{slot}(l) $$

With our implementation, a beacon size of $L_{beacon} = 3$ Bytes is sufficient. We use the values from Table I to derive $T_r$ as a function of the network diameter $H$ and the number of slots per round $B$. For example, in a 4-hop network using $5$-slot rounds, the minimum message latency is 50 ms (Fig.5).

We now consider the benefits of using rounds in terms of energy, using the radio-on time as a metric. Beacons are necessary to reliably prevent message collisions (Sec. II). In a design without rounds, each message transmission would be preceded by its own beacon, such that the transmission time for $B$ messages of size $l$ (denoted $T_{wo/r}(l)$) would take

Design, Automation And Test in Europe (DATE 2018)
Using (6), we compute the relative energy saving
\[ E = \frac{T_{\text{on}} - T_{\text{on}}}{T_{\text{on}}} \cdot \frac{T_{\text{on}} + T_{\text{slot}}}{T_{\text{on}}} \]
where \( T_{\text{on}} \) and \( T_{\text{slot}} \) are defined in (1) and (2), respectively. Therefore, we can expect that yes, an implementation of TTW will match the results of the performance modeling with high accuracy, thus validating TTW’s design for CPS applications.

V. RELATED WORK

Various high-reliability protocols have been proposed for low-power multi-hop wireless network, like TSCH [11], WirelessHART [9], LWB [10], and Blink [4]. Despite their respective benefits, those protocols consider only the network. They do not account for the schedule of distributed tasks, and therefore hardly support end-to-end deadlines, as commonly required for CPS [7]. In [5], a protocol that provides such end-to-end guarantees is proposed, but couples tasks and messages as loosely as possible, aiming for efficient support of sporadic or event-triggered applications. This results in high worst-case latency and is thus not suitable for time-critical CPS applications [7]. This points toward a fully time-triggered system where tasks and messages are co-scheduled.

In the wired domain, much work has been done on time-triggered architecture [21], the static-segment of FlexRay [22], or TTEthernet [23]. Many recent works use SMT or ILP methods to synthesize or analyze static schedules for those systems [24].

VI. CONCLUSIONS

This paper presented Time-Triggered Wireless (TTW), a distributed low-power wireless system design, motivated by the need for a wireless solution providing low latency, energy efficiency, runtime adaptability, and high reliability for time-critical applications, e.g., industrial closed-loop control systems. This paper introduced the design concepts that enable TTW to meet these requirements. The performance evaluation of TTW shows a reduction of communication latency by a factor 2× and of energy consumption by 33-40% compared to the closest related work [5]. This validates the suitability of TTW for wireless CPS applications and opens the way to its real-world implementation together with industry partners.

Acknowledgments. This work is supported by Nano-Tera.ch (Swiss Confederation financing), and by the German Research Foundation within cfai and SPP 1914, project EcoCPS.

REFERENCES


