Application-Aware Scheduling of Networked Applications over the Low-Power Wireless Bus

Kacper Wardega
Boston University
Boston, MA, USA
ktw@bu.edu

Wenchao Li
Boston University
Boston, MA, USA
wenchao@bu.edu

***Abstract*—Recent successes of wireless networked systems in advancing industrial automation and in spawning the Internet of Things paradigm motivate the adoption of wireless networked systems in current and future safety-critical applications. As reliability is key in safety-critical applications, in this work we present NETDAG, a scheduler design and implementation suitable for real-time applications in the wireless setting. NETDAG is built upon the Low-Power Wireless Bus, a high-performant communication abstraction for wireless networked systems, and enables system designers to directly schedule applications under specified task-level real-time constraints. Access to real-time primitives in the scheduler permits efficient design exploration of tradeoffs between power consumption and latency. Furthermore, NETDAG provides support for weakly hard real-time applications with deterministic guarantees, in addition to heretofore considered soft real-time applications with probabilistic guarantees. We propose novel abstraction techniques for reasoning about conjunctions of weakly hard constraints and show how such abstractions can be used to handle the significant scheduling difficulties brought on by networked components with weakly hard behaviors.**

***Index Terms*—application-aware scheduling, real-time scheduling, wireless networked systems, low-power wireless systems.**

I. INTRODUCTION

Across a wide swathe of engineering domains, it is often necessary for real-world systems to behave in a predictable manner. To this end, engineers employ models that help to link the passage of time with the evolution of the system under test, allowing designs to be validated for predictability. Systems designed in this fashion are generally called *real-time systems*, in the sense that the system was designed in a modeling framework that considers real-world timing aspects alongside the system evolution [1]. In the context of computer engineering, real-time system design is concerned with the real-time properties of computation. Commonly, it is stipulated that the latency of some computation have predictable behavior. Real-time models of computer systems rely heavily on assumptions made about the underlying hardware and software where the computation is occurring. As such, it is important to revisit real-time models as the hardware and software ecosystem changes.

Recent years have shown a growing trend towards *wireless networked systems* – that is, computer systems where independent compute-enabled components communicate via wireless medium. Wireless networked systems can be desirable over wired counterparts for several reasons: they may be cheaper to install and maintain, more flexible w.r.t. changing requirements, and amenable to mobile hardware components. From the viewpoint of real-time systems however, the wireless transmission of messages introduces significant uncertainties. In response, considerable effort has been made to study the real-time implications of de-wiring a system and to patch and update previous real-time models to extend to wireless networked systems. For example, mature real-time design methodologies for wireless networked systems leverage multi-channel TDMA for message transmission [2]. The primary shortcomings of existing techniques is a continued dependence on the particular network topology.

The Glossy flooding protocol shows promise as a foundation for the design of real-time applications on top of wireless networked systems in a topology-agnostic fashion [3]. Glossy floods are reliable, fast, and energy-efficient, while simultaneously providing all-to-all wireless communication and accurate clock synchronization. The authors of Glossy have long argued that the reliability of Glossy floods makes Glossy suitable for real-time applications, and have proposed a Low-Power Wireless Bus (LWB) abstraction on top of Glossy floods for enabling message-passing within wireless networked applications [4].

As individual Glossy floods have demonstrably high success rates, e.g. in excess of 99%, applications running over the LWB can be shown indirectly to have strong real-time performance. Our observation is that there is no current mechanism by which system designers can *directly schedule applications to adhere to pre-determined real-time constraints*. Another drawback of current approaches is the overwhelming focus on *probabilistic*, or soft, real-time properties of applications. The weakly hard paradigm [5], an alternative approach which focuses on a bounded non-deterministic viewpoint of real-time performance, has gone unstudied in the context of the LWB. In response, we claim the following contributions in this work:

- Improve upon the current state-of-the-art time-triggered LWB scheduler design by providing access to task-level soft and weakly hard real-time primitives.
- Provide the first openly-available implementation of a time-triggered LWB scheduler.
- We are the first to enable weakly hard real-time constraints on networked applications applications running over the LWB. We propose a novel abstraction technique

for layered weakly hard constraints and show how the layering abstraction can be used to schedule weakly hard real-time applications over the LWB.

II. Background

A. Low-power wireless bus

The LWB abstracts wireless communication and allows compute nodes to communicate with one another as if all of the nodes were connected by a single wired bus [4]. The LWB is *time-triggered*, and communication over the bus takes place in rounds. Each communication round consists of a *beacon* that conveys information about the current round, followed by any number of *contention-free slots* that contain application messages. Within a round, the beacon and each of the subsequent slots are transmitted in an all-to-all fashion by independent Glossy floods.

Glossy (see [3]) floods are themselves *event-triggered* as opposed to time-triggered. The beginning of a round triggers all of the nodes to turn on their radios in `RX` mode, sans the source node which begins by transmitting data, namely the message to be flooded, header and local clock information, and a relay counter (initially `0`). All nodes that successfully receive transmitted data immediately (triggered by radio event) increment the relay counter and switch to `TX` mode and begin re-transmission of the data. Nodes switch to `RX` mode each time they finish transmitting. Furthermore, nodes turn off their radios once the round is over (time-triggered) or once they have transmitted $N_{\texttt{TX}}$ times, a tunable parameter of the Glossy flood.

The event-triggered Glossy flood is reconciled with the time-triggered LWB through an estimate of how long the Glossy flood should take to complete. The estimate of flooding time is founded on hardware measurements of the radio wake-up time, delay, bit-rate, the software gap (of incrementing the relay counter), the width of the data, and finally a lower bound on the maximum relay counter. The maximum relay counter is a function of the bound on network diameter, $D(\mathcal{N})$, and $N_{\texttt{TX}}$.

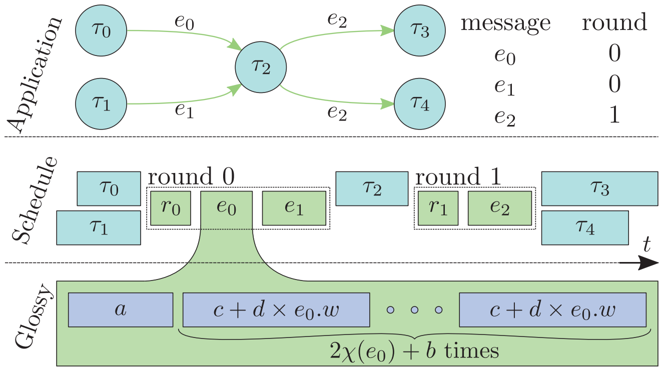

Fig. 1. NetDAG operates on applications that can be written as a task-dependency graphs (top). Messages that originate at the same node are treated as being the same. NetDAG schedules the application, taking into account not only the proper order of tasks and messages, but also the retransmission parameters of the Glossy floods responsible for message delivery (middle). The Glossy flood responsible for transmitting message e_0 (bottom) has retransmission parameter $\chi(e_0)$. If $\chi(e_0)$ is chosen too small, then τ_2, τ_3, τ_4 may not have the real-time behavior specified by the system designer. Too large, and the makespan becomes unnecessarily large.

The reason it is only a lower bound is because it assumes no failed transmissions – in our work we show how to tune $N_{\texttt{TX}}$ in order to provide real-time guarantees. A graphical view of a networked application running over the LWB is shown in fig. 1.

B. DAG scheduling

The input to a DAG scheduling problem for some application $\mathcal{A}$ is a labeled task-dependency graph of the application $G_{\mathcal{A}} = (T, \mathcal{E})$. Tasks (vertices) are labeled with WCETs and deadlines while messages (edges) are labeled with respective widths. In the classical formulation, the objective of the scheduler is to determine optimal task placement, e.g. on to available compute nodes, and task start times, given some additional information about task placement costs and message sending costs. In contrast, task placement is known for the typical wireless networked system, since tasks involve interactions with the real-world from a specific compute node. Therefore, in this work *the objective of the scheduler is to determine makespan-optimal task start times concurrently with assignments of messages to LWB rounds and the specific parameters of each underlying Glossy flood.*

C. Real-time applications

We consider two types of real-time constraints that designers may like to have access to when scheduling applications over a LWB. Task-level *soft* real-time constraints permit the system designer to stipulate that certain tasks must have success probability of at least some specified amount, given independent runs of the application. Soft real-time constraints are useful in situations where the quality-of-service of an application depends on the success rates of the constituent tasks, e.g. monitoring and sensing applications. Task-level *weakly hard* real-time constraints permit the system designer to stipulate the bounded non-determinism permitted of some task, (again) given independent runs of the application [5]. Weakly hard real-time constraints are useful in the specification of safety-critical or industrial control applications, as deterministic guarantees more readily lead to safety proofs. In lieu of probabilities, weakly hard constraints take the form of (m, K)-style constraints. A task-level constraint of (m, K) means that the task should succeed at least m times every K task executions. Similarly, a constraint of $\overline{(m, K)}$ means that the task should fail (due to missing dependencies) no more than m times every K executions. Formally, a k-sequence $\omega \in \{0,1\}^*$ models a weakly hard constraint (m, K), or $\omega \vdash (m, K)$, if for all indices t, $\sum_{i=t}^{t+K} \omega(i) \geq m$. Conversely, [5] defines *satisfaction sets* of (m, K) as $S^{\kappa}((m, K)) \overset{\text{def}}{=} \{\omega : \omega \text{ a } k\text{-sequence of length } \kappa \wedge \omega \vdash (m, K)\}$ and $S^{+}((m, K)) = \bigcup_{\kappa \in \mathbb{N}} S^{\kappa}((m, K))$. Table I compares soft and weakly hard constraints.

III. Application-Aware Scheduling

A. Communication-to-task transformation

Consider the problem of scheduling an application $\mathcal{A}$ consisting of finitely many tasks T on to a network $\mathcal{N} = (\mathcal{P}, \mathcal{C})$

TABLE I
COMPARISON BETWEEN SOFT AND WEAKLY HARD REAL-TIME CONSTRAINTS.

constraint	soft	weakly hard
guarantee	probabilistic	bounded non-deterministic
usage	monitoring/sensing	safety-critical/control
example	*task succeeds 84% of the time*	*task succeeds at least 6 times in every 10 consecutive executions*

with message-passing over a LWB. The dependency graph of $\mathcal{A}$ is given by a DAG $G_{\mathcal{A}} = (T, \mathcal{E})$. Each task is mapped by $\varrho : T \to \mathcal{P}$ to a physical computing device s.t. $\forall \tau, \mu \in T, p \in \mathcal{P}$

$$(\varrho(\tau) = p \wedge \varrho(\mu) = p) \Rightarrow \left(\tau \overset{G_{\mathcal{A}}}{\rightsquigarrow} \mu \vee \mu \overset{G_{\mathcal{A}}}{\rightsquigarrow} \tau\right) \quad (1)$$

Equation (1) is not strictly necessary, it just avoids the issue of solving the problem of placing non-interdependent tasks on the same compute node. The known WCET, or duration, of τ on $\varrho(\tau)$ is denoted $\tau.d \in \mathbb{N}$. As in prior work [4], we assume that edges with the same source carry the same information, since Glossy floods the message to all nodes. As such, we consider a restricted set $\mathcal{E}^* \subseteq \mathcal{E}$ of edges with unique source nodes. The width of a message $e \in \mathcal{E}^*$ is denoted $e.w \in \mathbb{N}$. Denote by $L(G_{\mathcal{A}})$ the line graph of $G_{\mathcal{A}}$. We define a *topological partial order* of an arbitrary line graph $G = (V, E)$ to be a function $l : V \to \mathbb{N}_{|V|}$ satisfying $\forall r, s \in V$

$$r \overset{G}{\rightsquigarrow} s \Rightarrow l(r) < l(s) \quad (2)$$

The definition of a valid topological partial order l of $L(G_{\mathcal{A}})$ assigns to each message a communication round, and allows us to define a predecessor operator $\mathrm{pred}(\tau) = \{x : x \in l(\mathcal{E}^*) \cup \mathcal{E}^*, x \overset{G_{\mathcal{A}}}{\rightsquigarrow} \tau\}$ for tasks $\tau \in T$. The duration $r.d$ of a communication round $r \in l(\mathcal{E}^*)$ is then given by

$$\begin{aligned} r.d = {} & \delta_r\left(a + (2\chi(r) + b)(c + d\gamma)\right) \\ & + \sum_{r = l(e)} a + (2\chi(e) + b)(c + d \times e.w) \end{aligned} \quad (3)$$

As mentioned in § II-A, (3) is just an estimate based on hardware profiling, message widths, and the best-case maximum relay counter. The a, b, c, d are known constants, $\chi(r), r \in l(\mathcal{E}^*)$ is the unknown Glossy parameter N_{TX} for the beacon of round r, $\chi(e), e \in \mathcal{E}^*$ is the unknown Glossy parameter N_{TX} for the slot containing message e, and δ_r is a binary variable that indicates if round r is non-empty.

We can now define a *feasible schedule* for $\mathcal{A}$ on $\mathcal{N}$ over the LWB as a tuple (ζ, χ, l), where $\zeta : T \cup l(\mathcal{E}^*) \to \mathbb{N}$, $\chi : \mathcal{E}^* \cup l(\mathcal{E}^*) \to \mathbb{N}_{>0}$ satisfy the conditions:

$$\begin{aligned} & \forall \tau, \mu \in T, \tau \overset{G_{\mathcal{A}}}{\rightsquigarrow} \mu \Rightarrow \zeta(\mu) - \mu.d > \zeta(\tau) \\ & \forall r, s \in l(\mathcal{E}^*), l(r) < l(s) \Rightarrow \zeta(s) - s.d > \zeta(r) \\ & \forall \tau \in T, e \in \mathcal{E}^*, (r = l(e)) \Rightarrow \\ & \Big(e \overset{G_{\mathcal{A}}}{\rightsquigarrow} \tau \Rightarrow \zeta(\tau) - \tau.d > \zeta(r) \\ & \wedge \tau \overset{G_{\mathcal{A}}}{\rightsquigarrow} e \Rightarrow \zeta(r) - r.d > \zeta(\tau)\Big) \end{aligned} \quad (4)$$

that tasks and communication rounds occur in the correct order,

$$\begin{aligned} & \forall r \in l(\mathcal{E}^*), x \in T \cup l(\mathcal{E}^*) \setminus r, \\ & (\zeta(x) < \zeta(r) - r.d) \vee (\zeta(x) - x.d > \zeta(r)) \end{aligned} \quad (5)$$

and that no task occurs at the same time as any communication.

In plain English, ζ sets the deadlines of the tasks and communication rounds, while χ tunes the number of retransmissions (N_{TX} parameter) of each message slot and round beacon. Our scheduler design admits only those applications where the task-dependency graph and task/message properties such as duration and width are known; as such, our scheduler design is termed *application-aware*. Table II contains a glossary of all symbols used throughout the text.

B. Soft real-time applications

For soft real-time applications, the system designer has probabilistic constraints $\mathcal{F}_{\mathrm{S}} : T \to [0, 1)$ selecting the desired minimum probability that a task succeeds given independent runs of $\mathcal{A}$. Note that $\mathcal{F}_{\mathrm{S}}$ has structure dictated by $G_{\mathcal{A}}$, namely that $\forall \tau, \mu \in T$, $\tau \overset{G_{\mathcal{A}}}{\rightsquigarrow} \mu \Rightarrow \mathcal{F}_{\mathrm{S}}(\tau) > \mathcal{F}_{\mathrm{S}}(\mu)$. The reason for the strict inequality on the RHS is that since messages are passed over the LWB, there is no way to *ensure* 100% transmission performance. We assume that the designer knows a priori the success rate of a Glossy flood as a function of the parameter N_{TX}, i.e. the success rate of message sending is a monotonically increasing function $\lambda_{\mathrm{S}} : \mathbb{N} \to [0, 1)$. Since flooding failures are independent, we can write the satisfaction of $\mathcal{F}_{\mathrm{S}}(\tau)$ as

$$\mathcal{F}_{\mathrm{S}}(\tau) \leq \Pi_{x \in \mathrm{pred}(\tau)} \lambda_{\mathrm{S}}(\chi(x)) \quad (6)$$

A feasible schedule that also satisfies $\mathcal{F}_{\mathrm{S}}$ is called a *feasible soft real-time schedule*. We have implemented makespan-optimal soft real-time scheduling through both MILP and SMT encodings [6], [7].

TABLE II
GLOSSARY OF SYMBOLS AND NOTATIONS.

$D(G)$	diameter of graph G
N_{TX}	retransmission parameter of a Glossy flood
$\mathcal{A}, G_{\mathcal{A}} = (T, \mathcal{E})$	an application and its task-dependency graph
$\mathcal{E}^*$	messages with unique senders
$S^{\kappa}((m, K))$, $S^{+}((m, K))$	κ-length sequences, sequences that satisfy (m, K)
$\mathcal{N} = (\mathcal{P}, \mathcal{C})$	physical network
ρ	mapping of tasks to physical nodes
$L(G)$	line graph of graph G
$\overset{G}{\rightsquigarrow}$	order induced by DAG G
δ_r	binary symbolic variable indicating round r is non-empty
a, b, c, d	known Glossy flood parameters
$\chi(e), \chi(r)$	symbolic variable specifying $N_{\mathrm{TX}}$for the Glossy flood for message e, for beacon of round r
l	topological partial order
$\mathrm{pred}(\tau)$	set of rounds and tasks that precede τ
$\zeta(\tau), \zeta(r)$	deadline of task τ, round r
$\lambda_{\mathrm{S}}, \lambda_{\mathrm{WH}}$	soft, weakly hard network statistic
$\mathcal{F}_{\mathrm{S}}, \mathcal{F}_{\mathrm{WH}}$	task-level soft, weakly hard constraints
$[(m, K)]$	equality class of (m, K) induced by $\preceq$
$\oplus$	min-plus abstraction for conjunction of weakly hard constraints

C. Weakly hard real-time applications

A drawback of the soft real-time paradigm is that it can only provide system designers with probabilistic guarantees, which may not be strong enough for safety-critical applications. To this end, the system designer can model communication performance not with probabilities, but with some notion of bounded non-determinism. In particular, the designer assigns to each task (m, K)-style constraints $\mathcal{F}_{\mathtt{WH}} : T \to \mathbb{N}^2$ s.t. $\forall (m, K) \in \mathcal{F}_{\mathtt{WH}}(T)$, $m \leq K \wedge m > 0$. As in the soft real-time case, $\mathcal{F}_{\mathtt{WH}}$ inherits some structure from $G_{\mathcal{A}}$. In the weakly hard case, we have that $\forall \tau, \mu \in T$, $\tau \overset{G_{\mathcal{A}}}{\rightsquigarrow} \mu \Rightarrow \mathcal{F}_{\mathtt{WH}}(\tau) \preceq \mathcal{F}_{\mathtt{WH}}(\mu)$. Here, $\preceq$ is a partial order (due to [5]) on weakly hard constraints defined by the property

$$(\alpha, \beta) \preceq (\gamma, \delta) \Leftrightarrow \\ \gamma \leq \max\left\{ \left\lfloor \frac{\delta}{\beta} \right\rfloor \alpha, \delta + \left\lceil \frac{\delta}{\beta} \right\rceil (\alpha - \beta) \right\} \quad (7)$$

The designer in the weakly hard case knows the bounded failure behavior of a Glossy flood as a function of the $N_{\mathtt{TX}}$ parameter, $\lambda_{\mathtt{WH}} : \mathbb{N} \to \mathbb{N}^2$, where $\lambda_{\mathtt{WH}}$ satisfies the same co-domain properties as $\mathcal{F}_{\mathtt{WH}}$ and is monotonically increasing w.r.t. $\preceq$. Satisfaction of $\mathcal{F}_{\mathtt{WH}}$ is significantly more difficult to write down than for $\mathcal{F}_{\mathtt{S}}$ due to the combinatorial nature of bounded non-determinism. As in (6), all of the Glossy floods on which τ depends must succeed in order for τ to succeed. Since we now have bounded non-deterministic behavior of the Glossy floods, checking satisfaction of $\mathcal{F}_{\mathtt{WH}}(\tau)$ involves checking if for all k-sequences $\omega_i \in S^+(\chi(i))$, $i \in \mathrm{pred}(\tau)$, it is the case that the k-sequence ω, $\omega \overset{\text{def}}{=} \wedge_i \omega_i$, is a model of $\chi(\tau)$. To reduce the complexity of solving universally quantified equations, we introduce the first, to the best of our knowledge, abstraction for *layered*, or rather conjunctions of, weakly hard constraints.

Given two weakly hard constraints x and y, we define the set $\Omega^{\oplus}(x, y) = \{[z] : \omega_l \in S^+(x) \wedge \omega_r \in S^+(y) \Rightarrow (\omega_l \wedge \omega_r) \in S^+(z)\}$. The set $\Omega^{\oplus}(x, y)$ contains the equality classes $[z]$ of weakly hard constraints that are guaranteed to hold under any conjunction of two k-sequences ω_l and ω_r that satisfy x and y respectively. We define our abstraction of the conjunction of weakly hard constraints by

$$\overline{(\alpha, \gamma)} \oplus \overline{(\beta, \delta)} \overset{\text{def}}{=} \overline{(\min\{\alpha + \beta, \gamma, \delta\}, \min\{\gamma, \delta\})} \quad (8)$$

At first glance, (8) may seem disappointingly simple as it states that the conjunction of two weakly hard constraints results in a new weakly hard constraint that adds the number of misses allowed, restricted to the smaller of the two windows γ, δ. Formally, we state the following results.

Soundness

Given two weakly hard constraints x and y, $\exists [z] \in \Omega^{\oplus}(x, y)$ s.t. $x \oplus y \in [z]$.

Tightness

There exist infinitely many pairs of weakly hard constraints x and y s.t. $x \oplus y \in \inf \Omega^{\oplus}(x, y)$.

Proof Given weakly hard constraints $\overline{(\alpha, \gamma)}$ and $\overline{(\beta, \delta)}$, let $\omega_l \in S^+(\overline{(\alpha, \gamma)})$ and $\omega_r \in S^+(\overline{(\beta, \delta)})$. Assume WLOG that $\gamma \leq \delta$ (since $\oplus$ commutes), then $\omega_r \in S^+(\overline{(\beta, \gamma)})$ since by corollary of (7), $\overline{(\beta, \delta)} \preceq \overline{(\beta, \gamma)}$. Now consider any γ-length window of $\omega_l \wedge \omega_r$, the worst case number of misses is clearly $\min\{\alpha + \beta, \gamma\}$; the number of misses can get no worse than by having all α misses allowed in ω_l followed by all β misses allowed in ω_r. Incidentally, $\oplus$ is tight whenever $\gamma = \delta$. ■

For the price of completeness, the $\oplus$ abstraction for conjunction of weakly hard constraints allows us to write the satisfaction of $\mathcal{F}_{\mathtt{WH}}(\tau)$, without universal quantifiers, as

$$\bigoplus_{x \in \mathrm{pred}(\tau)} \lambda_{\mathtt{WH}}(\chi(x)) \preceq \mathcal{F}_{\mathtt{WH}}(\tau) \quad (9)$$

Equation (9) is not amenable to disciplined quasi-convex programming rules, which precludes the use of MILP solvers [8]. On the other hand, there is currently no SMT solver for a theory including $\lfloor \cdot \rfloor$, $\lceil \cdot \rceil$ functions. As a result, we again apply corollaries of (7) to obtain our final abstraction for checking the satisfaction of $\mathcal{F}_{\mathtt{WH}}(\tau)$:

$$\left(\bigoplus_{x \in \mathrm{pred}(\tau)} \lambda_{\mathtt{WH}}(\chi(x)) \right).m \geq \mathcal{F}_{\mathtt{WH}}(\tau).m \\ \wedge \left(\bigoplus_{x \in \mathrm{pred}(\tau)} \lambda_{\mathtt{WH}}(\chi(x)) \right).K \leq \mathcal{F}_{\mathtt{WH}}(\tau).K \quad (10)$$

A feasible schedule that also satisfies $\mathcal{F}_{\mathtt{WH}}$ is called a *feasible weakly hard real-time schedule*. We implement weakly hard real-time scheduling through an encoding to SMT, which provides makespan-minimal schedules subject to the (10) abstraction.

IV. Experiments

A. Validation

We perform a simulation-based validation (sanity-check) of our scheduling formulation. Given a real-time schedule (ζ, χ, l) for an application $\mathcal{A}$, we check the real-time properties of a task τ of $\mathcal{A}$ given independent runs of the system. To simulate κ runs of τ, we sample k-sequences ω_x of length κ at random from each predecessor $x \in \mathrm{pred}(\tau)$ of τ w.r.t. to the network statistic. The behavior of τ is then given by $\omega_\tau(t) = \wedge_x \omega_x(t)$. In the soft real-time case,

$$\omega_x(t) \overset{\text{i.i.d.}}{\sim} \text{Bernoulli}(\lambda_{\mathtt{S}}(\chi(x))) \quad (11)$$

The test statistic is given by $\upsilon = \sum_t \omega_\tau(t)/\kappa$, and can be used to construct a test for $\upsilon \geq \mathcal{F}_{\mathtt{S}}(\tau)$. In the weakly hard real-time case, we obtain interesting miss-patterns for task x by synthesizing sequences that are in the satisfaction set of $\lambda_{\mathtt{WH}}(\chi(x))$, but not in the satisfaction set of some weaker constraints. Namely, say that $\lambda_{\mathtt{WH}}(\chi(x)) = (m_x, K_x)$, then we synthesize

$$\omega_x \in S^\kappa((m_x, K_x)) \\ - S^\kappa((m_x - 1, K_x)) \\ - S^\kappa((m_x, K_x + 1)) \quad (12)$$

Here, we just check to ensure that $\omega_\tau \vdash \mathcal{F}_{\mathtt{WH}}(\tau)$. Our scheduling implementation and validation scripts are available at https://github.com/netdag/netdag.

B. Wireless MIMO & switched controllers

Our NETDAG approach can be gracefully applied to schedule safety-critical applications with multiple controllers, for example *multi-input/multi-output* (MIMO) or *switched control* applications (or applications that are both). MIMO applications are those with multiple control tasks, each of which receives input messages from sensing tasks and emits messages to different sets of actuation tasks. Designers can leverage our scheduler to freely configure how often each control output is required (and by which actuation task), and NETDAG will minimize the amount of time spent on communication. Switched control applications are those with multiple control tasks that each message the same actuation task – some of the control tasks may offer a higher quality of control output at the expense of a larger WCET. Exactly as in the MIMO situation, system designers can simply specify how often each type of control output is required and again the scheduler will take care of reorganizing communications in an optimal fashion.

As a demonstration, we schedule the simple MIMO application $\mathcal{A}_{\text{MIMO}}$ consisting of six sensing tasks, three control tasks, and four actuator tasks, and randomly selected links between task sets. We assign the underlying network $\mathcal{N}$ a synthetic weakly hard network statistic:

$$\lambda(n) = \overline{\left(\left\lfloor 10e^{-\frac{1}{2}n}\right\rfloor + 1, 20n\right)} \tag{13}$$

Equation (13) is a valid weakly hard network statistic, as it satisfies $\forall n, k \in \mathbb{N}_{>0}, n < k \Rightarrow \lambda(k) \preceq \lambda(n)$. The system designer can then query the NETDAG scheduler for the minimum feasible latency for $\mathcal{A}_{\text{MIMO}}$ under a variety of task-level weakly hard constraints. In our experiment, we incrementally assign weakly hard constraints to the actuation tasks. Fig. 2 reports our scheduler's output, showing how application latency increases as weakly hard constraints become stricter and as more actuation tasks have weakly hard constraints.

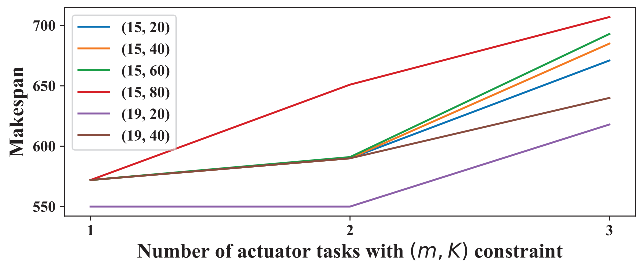

Fig. 2. The makespan of a MIMO application increases as weakly hard constraints are incrementally applied to the actuator tasks. The makespan also increases as the applied weakly hard constraint becomes more strict.

C. Weakly hard constraints for control tasks

In [9], the authors demonstrate high-performance wireless control in a cartpole environment over the LWB. The authors demonstrate the tradeoff between the amount of time spent in communication and the average availability of the current control output at the actuating node. Unfortunately, it is often difficult to reason about the safety or even stability of a controlled system using only the average control output availability. In [10], the authors show that it is possible to prove forward reachability for some dynamical systems if the behavior of the controller output is weakly hard, motivating the usage of weakly hard constraints as a design methodology for safety-critical systems.

In order to better understand empirically how weakly hard behaviors impact control performance in dynamical systems, we inject faults in the control of a classic cartpole environment. The weakly hard miss-patterns ω are sampled according to (12). The controller c is a state-of-the-art neural network that maps inputs in the state space X to control outputs Y. The control signal y is then given in terms of the observation signal x,

$$y(t) = (1 - \omega(t))c(x_t) + \omega(t)y(t-1) \tag{14}$$

for time $t > 0$ and initial condition $y_0 = 0$. As expected, we observe that under fixed K, increasing the number of permitted misses m lowers the average performance of the controller. Conversely, under fixed m, increasing the window size K increases average performance. These trends are highlighted in fig. 3.

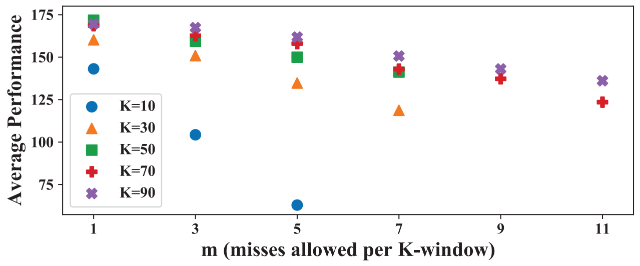

Fig. 3. Impact of changing weakly hard behavior of state-of-the-art neural network controller on cartpole balance performance. Performance is measured in number of time steps on average that the controller can keep the cartpole balanced over a set of injected (m, K) faults.

D. Design space exploration

In the design of *low-power* real-time wireless networked systems, system designers will like to know what options are available as far as power consumption of the inter-node communication layer. NETDAG offers designers the ability to efficiently explore the impact of different radio configurations on the real-time performance of an application. Changing the transmission power of the radios across Q_i different power settings results in different bounds on the network diameter, $(D(\mathcal{N}))_i$, and network statistics λ_i. The system designer, equipped with the $(D(\mathcal{N}))_i$, λ_i, can then leverage NETDAG to discover the minimal transmission power setting Q_i that satisfies the application's task-level deadline constraints.

We perform a proof-of-concept experiment that illustrates the transmission power exploration workflow by simulating an application that is to run on physical nodes that are mobile within the unit square. Each node has a finite number of transmission power settings $Q_i \in Q = (0, 1]$. We define the pairwise signal strength between nodes $x, y \in P$ separated by radius $r(x,y)$ as $\texttt{SS}_i(x,y) = Q_i / r(x,y)^2$. Signal strength saturates at 2 and nodes with pairwise signal strength at or below 0.5 are out-of-range. We denote by $\texttt{fSS}_i$ the saturation- and out-of-bound- filtered signal strength function with co-domain $(0.5, 2]$. As the nodes move in the unit square, the system designer profiles the worst-case average (over pairs of nodes) pairwise signal strength, $\overline{\texttt{fSS}_i}$, and network diameter, $(D(\mathcal{N}))_i$, against the transmission power Q_i. The results of profiling in-simulation are shown in the left two plots of fig. 4. We pick a soft real-time statistic parameterized by $\overline{\texttt{fSS}_i}$,

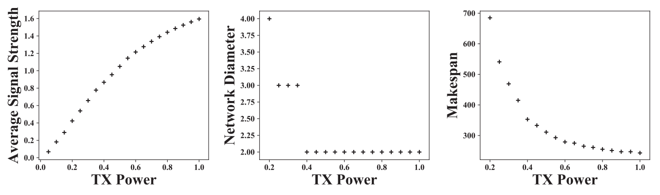

Fig. 4. Transmission power design exploration workflow. The designer begins by profiling the network statistic and network diameter under different $\texttt{TX}$ power settings Q_i. NETDAG can then provide the changing real-time performance of an application running atop the wireless networked system.

$$\lambda_i(n) = \frac{2}{1 + e^{-\overline{\texttt{fSS}_i} \cdot n}} - 1 \tag{15}$$

Equation (15) satisfies the required network statistic properties since it has co-domain $[0, 1)$ and is monotonically increasing for $n \geq 1$. Finally, we can feed λ_i and $(D(\mathcal{N}))_i$ to NETDAG and find the end-to-end latency of some application $\mathcal{A}$. The right plot of fig. 4 shows the end-to-end latency of $\mathcal{A}_{\text{MIMO}}$ as a function of Q_i.

V. Related Work

The formulation of the DAG scheduling problem over the LWB is largely the same as the one found in [11]; in our work we do not consider the additional complexities of operating modes. Similarly motivated by exploring the tradeoff between control performance and communication overhead, other scholarship has also considered modulating the communication overhead for LWB-based networked applications under changing requirements on the availability of data sent over the LWB, namely [12] proposes efficient methods for modulating application period at runtime for control applications. Orthogonal to our work is the enabling of dynamic scheduling for real-time LWB-based applications. Blink is a novel dynamic real-time wireless protocol over the LWB [13], where the main result is a *contract and guarantee* system for applications wishing to send messages over the LWB. The authors of [14] perform extensive real-time analysis of the entire transmission chain to prove real-time properties of Blink. A key result that establishes a strong theoretical foundation for our work (and for [11], [13], [14]) is [15], in which the authors show that Glossy floods can be treated as independent Bernoulli trials and demonstrate that this statistical fact can be used by a scheduler to provide soft real-time guarantees. Inspired by [16], ours is the first work to consider weakly hard constraints on wireless applications over the LWB.

VI. Conclusion

We have presented NETDAG, a time-triggered scheduler design for real-time wireless networked systems. NETDAG is not only the first such scheduler with support for the weakly hard real-time paradigm, but also the first LWB-based scheduler that directly supports real-time constraints in the form of task-level soft or weakly hard primitives. The weakly hard paradigm is a useful tool for dealing with uncertainties stemming from wireless communication, and we have introduced an abstraction to ease the scheduling of networked weakly hard real-time applications. Furthermore, we have shown that communication reliability is an important parameter in the design of real-time wireless networked systems and that it can be used to explore the energy/performance trade-off. We believe that enabling the weakly hard paradigm is an important step towards reliable real-time networked applications.

References

[1] K. G. Shin and P. Ramanathan. "Real-time computing: a new discipline of computer science and engineering". *Proceedings of the IEEE*, vol. 82, no. 1, pp. 6–24, 1994.

[2] J. Song, et al. "WirelessHART: Applying wireless technology in real-time industrial process control". *Proceedings of the IEEE Real-Time and Embedded Technology and Applications Symposium (RTAS)*, pp. 377–386, 2008.

[3] F. Ferrari, M. Zimmerling, L. Thiele, and O. Saukh. "Efficient network flooding and time synchronization with glossy". *Proceedings of the ACM/IEEE International Conference on Information Processing in Sensor Networks (IPSN)*, pp. 73–84, 2011.

[4] F. Ferrari, M. Zimmerling, L. Mottola, and L. Thiele. "Low-power wireless bus". *Proceedings of the 10th ACM Conference on Embedded Network Sensor Systems (SenSys)*, p. 1, 2012.

[5] G. Bernat, A. Burns, and S. Member. "Weakly hard real-time systems". *IEEE Transactions on Computers*, vol. 50, no. 4, pp. 308–321, 2001.

[6] L. De Moura and N. Bjørner. "Z3: An efficient SMT Solver". *Lecture Notes in Computer Science*, vol. 4963, pp. 337–340, 2008.

[7] Gurobi Optimization, LLC. "Gurobi optimizer reference manual", 2019.

[8] A. Agrawal and S. Boyd. "Disciplined quasiconvex programming". arXiv:1905.00562, 2019.

[9] F. Mager, et al. "Feedback control goes wireless: guaranteed stability over low-power multi-hop networks". *Proceedings of the 10th ACM/IEEE International Conference on Cyber-Physical Systems (ICCPS)*, 2019.

[10] C. Huang, W. Li, and Q. Zhu. "Formal verification of weakly-hard systems". *Proceedings of the 22nd ACM International Conference on Hybrid Systems: Computation and Control (HSCC)*, pp. 197–207, 2019.

[11] R. Jacob, et al. "TTW: a time-triggered wireless design for CPS". *Proceedings of the Design, Automation and Test in Europe Conference and Exhibition (DATE)*, pp. 865–868, 2018.

[12] Y. Ma and C. Lu. "Efficient holistic control over industrial wireless sensor-actuator networks". *Proceedings - IEEE International Conference on Industrial Internet (ICII)*, pp. 89–98, 2018.

[13] M. Zimmerling, L. Mottola, P. Kumar, F. Ferrari, and L. Thiele. "Adaptive real-time communication for wireless cyber-physical systems". *ACM Transactions on Cyber-Physical Systems*, vol. 1, no. 2, pp. 1–29, 2017.

[14] R. Jacob, M. Zimmerling, P. Huang, J. Beutel, and L. Thiele. "End-to-end real-time guarantees in wireless cyber-physical systems". *Proceedings - Real-Time Systems Symposium*, pp. 167–178, 2017.

[15] M. Zimmerling, F. Ferrari, L. Mottola, and L. Thiele. "On modeling low-power wireless protocols based on synchronous packet transmissions". *Proceedings - IEEE Computer Society's Annual International Symposium on Modeling, Analysis, and Simulation of Computer and Telecommunications Systems, MASCOTS*, pp. 546–555, 2013.

[16] C. Huang, K. Wardega, W. Li, and Q. Zhu. "Exploring weakly-hard paradigm for networked systems". *Proceedings of the Workshop on Design Automation for CPS and IoT*, pp. 51–59, 2019.