Towards Real-time Wireless Cyber-physical Systems

Romain Jacob* Marco Zimmerling† Pengcheng Huang* Jan Beutel* Lothar Thiele*
*Computer Engineering and Networks Laboratory, ETH Zurich, Switzerland
†Networked Embedded Systems Group, TU Dresden, Germany
firstname.lastname@tik.ee.ethz.ch marco.zimmerling@tu-dresden.de

***Abstract*—One big challenge to be overcome before the successful deployment of wireless cyber-physical systems is to provide *hard real-time guarantees*, not only within the wireless network, but in fact between *end-to-end application processes*. To achieve this, we design a distributed real-time protocol (DRP) that considers the complete transmission chain, including application tasks, peripheral busses, memory accesses, networking interfaces, and wireless real-time protocol. DRP guarantees that end-to-end message deadlines are met, while being adaptive to unpredictable system changes, by establishing at run-time a set of contracts among the different elements of the transmission chain.**

I. Introduction

Over the past decade, a tremendous amount of work has been carried out around low-power wireless communication technologies. Especially, wireless sensor networks (WSNs) have received much attention. One major challenge yet to be overcome is to enable reliable and efficient use of low-power wireless networking Cyber-Physical Systems (CPS), sometimes referred to as wireless sensor and actuator networks. As many CPS applications are mission-critical and physical processes evolve as a function of time, the communication among the sensing, actuating, and computing elements in CPS is often subject to real-time requirements (*e.g.*, to guarantee the stability of feedback loops). Such real-time requirements are key to enable *safe* CPS, which is arguably one of the most important features for a successful deployment of CPS [1].

Challenges. These real-time requirements are often specified from an *end-to-end application* perspective. For example, a control engineer may require that sensor readings taken at time t_s are available for computing the control law at $t_s + \mathbf{D}$. Here, the *relative end-to-end deadline* $\mathbf{D}$ is derived from the activation times of *application-level tasks*, namely the sensing and control tasks, which are typically executed on physically distributed devices. Meeting such hard end-to-end deadlines is non-trivial, because data transfer between application-level tasks involves multiple other tasks (*e.g.*, operating system, networking protocols) and shared resources (*e.g.*, memories, system busses, wireless medium). Therefore, the entire transmission chain, involving application tasks, peripheral busses, memory accesses, and wireless networking protocol, must be taken into account to tackle this challenge. We argue that doing so requires combining three building blocks:

1) on the node level, a decoupling of application (*e.g.*, sensing, actuation, control) and wireless communication tasks using a *dual-processor architecture*;
2) on the network level, an efficient *wireless real-time protocol*, which guarantees that messages between source and destination nodes are delivered reliably in real-time;
3) overall, a *distributed real-time protocol* that manages the flows of messages across the network, decouples responsibilities between components, and ensures that end-to-end deadlines *between application tasks* are met.

In this paper, we first briefly present the design of our solution (Sec. II), then we highlight remaining problems and how we intend to address them in our ongoing work (Sec. III).

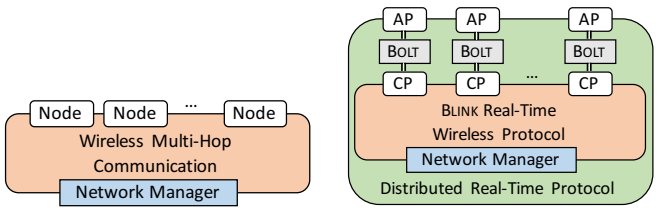
II. Overview of the solution

We consider that all communications between a source and destination node are subject to real-time constraints. We call one such communication a *flow*. Let $\mathcal{F}$ be the set of all real-time message *flows* in the system. Each flow $F_i = (s_i, d_i, T_i, \mathbf{D}_i)$ is defined by a *source node* s_i, that *releases* messages with a *minimum inter-message interval* T_i, also called *period*. Every message released by s_i should be delivered to the *destination node* d_i within the same *relative end-to-end deadline* $\mathbf{D}_i$. No message can be sent outside of a flow, and each flow must be *registered* (*i.e.*, accepted by the protocol) before it can start releasing messages. Several flows, eventually with different period and deadline, may be registered between the same source and destination nodes.

The wireless network consists of a set of *nodes* $\mathcal{N}$ that exchange messages via *wireless multi-hop communication*, as illustrated in Fig. 1(a). A logically global *network manager* arbitrates access to the shared networking resource. Physically, the network manager may run on one of the nodes. The overall design is based on three building blocks, as described next.

Dual-processor architecture. When using the traditional system architecture shown in Fig. 1(a), application and communication tasks execute concurrently on each node. When both tasks attempt to simultaneously access shared resources (*e.g.*, memory, processor, system bus), one of them will be delayed for an arbitrary time. Such interference hampers timing predictability, making real-time guarantees difficult to provide.

To tackle this issue, we propose to replace each node with a dual-processor platform. One processor (*AP*) runs only the application, while the other processor (*CP*) executes only the wireless multi-hop communication protocol. Using the Bolt interconnect [2], AP and CP are decoupled in time, power, and clock domains, and can asynchronously exchange messages with bounded delay, with ultra-low-power overhead.



(a) A set of nodes, each with a single processor, execute the application and exchange messages via wireless multi-hop communication.

(b) Application (AP) and communication (CP) processors exchange messages via Bolt, while the CPs run the Blink real-time wireless protocol.

Figure 1. Traditional (a) and our proposed (b) system architecture. *A logically global network manager arbitrates access to the shared wireless medium.*

Wireless real-time protocol. Providing real-time guarantees across wireless networks is challenging. In particular, to support real-world CPS applications, one needs a protocol that delivers packets reliably (*i.e.*, with high reception rate) within real-time deadlines, while being adaptive to dynamic changes in the wireless network and traffic demands.

Out of the many solutions that have been proposed over the years, Blink [3] is the only wireless real-time protocol that satisfies such requirements. It is built on top of LWB [4] and leverages fast and highly reliable Glossy floods [5], a protocol based on synchronous transmissions, which represents a paradigm shift away from traditional routing-based protocols.

Distributed real-time protocol. The use of Bolt to decouple the application (running on the APs) and the Blink wireless real-time protocol (running on the CPs) brings necessary benefits, such as flexibility in the operation mode for each component (*e.g.*, time- versus event-triggered) and interference mitigation. But at the same time, it also triggers a significant challenge: while the APs and CPs should execute independently to prevent interference, it is their *joint* operation that determines whether messages exchanged between the APs meet their end-to-end deadlines.

To address this challenge, we introduce a distributed real-time protocol (DRP), which represents the third building block of our solution. DRP strikes a balance between the decoupling of AP and CP/Blink on the one hand and the end-to-end timing predictability of message exchanges between the source and destination APs of one flow on the other hand. This trade-off is embodied by mutual *contracts*. A contract settles the least required agreement between APs, CPs and Blink such that all can operate as much as possible independently, while ensuring that end-to-end message deadlines are met. DRP establishes contracts at runtime, as flows are dynamically requested and removed, and scales efficiently to large sets of real-time message flows.

Using these three building blocks, the overall system architecture evolves from Fig. 1(a) to Fig. 1(b). In the remainder of this section, we detail the design of DRP and how contracts are established when a new flow is requested at runtime.

DRP: Distributed Real-time Protocol

DRP uses contracts that are dynamically established at runtime to provide end-to-end real-time communication between the source and destination node's AP of every flow. This includes guaranteeing that message buffers along the whole transmission chain never overflow.

In a nutshell, the overall end-to-end latency of a message depends on how often APs and CPs read out messages from Bolt, and the maximal delay for a message to be served by Blink. This can be formalized by three parameters for each flow F_i: the *flushing periods of the source* and *destination* nodes, and the *network deadline* of F_i, denoted by T_f^s, T_f^d, and D_i respectively. The network deadline D_i is computed online by the source node's AP when a new flow is requested. It represents the deadline which is requested to Blink (*i.e.*, if Blink accepts this new flow, it guarantees that any message is delivered at the destination node's CP within D_i).

DRP decides on the *distribution of responsibilities* among the source node, Blink, and the destination node of a flow F_i with regard to meeting its end-to-end deadline $\mathbf{D}_i$ using a configuration parameter of DRP, the *deadline ratio* r, chosen at design time. The source node and Blink are jointly responsible for meeting a fraction r of the end-to-end deadline $\mathbf{D}_i$,

$$f(T_f^s, D_i) = r * \mathbf{D}_i \tag{1}$$

The remaining part of the overall end-to-end deadline determines the responsibility of the destination node,

$$g(T_f^d) = (1 - r) * \mathbf{D}_i \tag{2}$$

One can derive concrete expressions for $f(\cdot,\cdot)$ and $g(\cdot)$ after a detailed worst-case timing and buffer analysis of the system.

Overall, DRP dynamically establishes two contracts for each newly admitted flow $F_i \in \mathcal{F}$ in the system:

Source $\leftrightarrow$ Blink F_i's source node s_i agrees to write no more messages than specified by the flow period T_i, and prevents overflows of Bolt and CP_s's local message buffer. In turn, Blink agrees to serve F_i such that any received message meets the network deadline D_i.

Blink $\leftrightarrow$ Destination Blink agrees to deliver no more messages than specified by T_i. In turn, F_i's destination node d_i agrees to read out all delivered messages such that overflows of Bolt and CP_d's local buffer are prevented and all messages meet the flow's end-to-end deadline $\mathbf{D}_i$.

For any flow, if both contracts are fulfilled, all messages that are successfully delivered by Blink will meet their end-to-end deadlines. In practice, the fulfillment of contracts is guaranteed by a set of *admission tests*, which are performed in sequence upon registration of a new flow. The overall mechanism of contracts (*i.e.*, sharing of responsibility, flow registration, and admission tests) is illustrated on Fig. 2.

In our ongoing work, we have derived the theoretical optimal performances that can be provided by DRP, in terms of responsiveness (*i.e.*, smallest end-to-end deadline) and bandwidth that can be supported. We also evaluated the run-time behavior of DRP in simulation, based on values and parameters from physical implementations of both Bolt and Blink. Our results show that the end-to-end latency of messages can be up to 96% of the our analytic worst-case

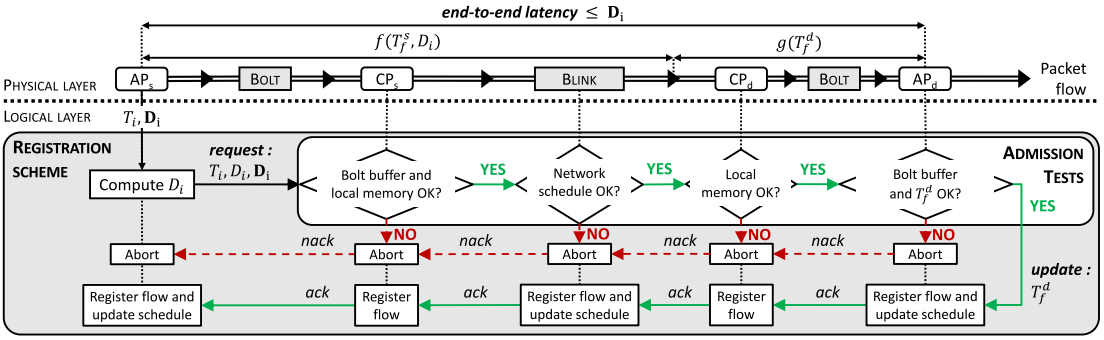


Figure 2. Steps and components involved when registering a new flow in DRP. *Given a request for a new flow $F_i = (s_i, d_i, T_i, \mathbf{D}_i)$, the source node's AP computes the flow's network deadline D_i. Then, all components successively verify using specific admission tests whether they can admit the new flow. DRP registers a flow only if all admission tests succeed, which triggers changes in the runtime operation (i.e., schedule) of some components.*

bounds. Thus, since our model of DRP's performance is safe and tight, it can be leveraged for system design.

III. Open Questions and Future Work

As we mention in the introduction, DRP is an initial step, and requires further investigations before we can claim it efficiently solves the real-time challenge in wireless CPS. In particular, we detail thereafter three key open questions that we intend to investigate in future work.

Physical implementation of DRP. Even though our preliminary simulation results are encouraging, they are not sufficient to validate the suitability of the protocol in a real-life setting. As illustrated in Fig. 1(b), DRP requires specific hardware (*i.e.*, a Bolt-enabled wireless network). Our group recently designed and produced custom-built dual-processor boards where Bolt interconnects a powerful application processor (TI MSP432) with a state-of-the-art communication processor (TI CC430). These have been implemented on the Flocklab testbed [6] and this new network will be publicly available for testing soon.

Leveraging this experimental setting, we plan to implement DRP and extensively test it to validate the registration and de-registration of flows at run-time, experiment with the failure and recovery of nodes, and verify that end-to-end real-time guarantees hold in such dynamic scenarios. The accurate time synchronization of FlockLab will allow us to validate the analytic worst-case delay analysis in a real system.

Dependability. Blink is built on the state-of-art wireless protocol LWB which achieves more than 99.9% data yield [4]. However, packet loss may still happen, and the sensitivity of DRP to these losses must be investigated. While a moderate loss of data packets can often be tolerated by the application, loss of schedule packets, that drive the operation of Blink, may be more critical.

We need now to develop a retransmission and/or dependability scheme for DRP to mitigate such effects and provide (at least) probabilistic guarantees for a safe behavior of the overall protocol, using *e.g.*, probabilistic model-checking. Some inspiration may be found in [7].

Reaching meaningful performances. Finally, the goal of this work is to enable practical implementations of wireless CPS. To this end, DRP must meet relevant latency requirements (*e.g.*, as mentioned in [1]). This means flow periods and end-to-end deadlines ranging from tens of milliseconds to half a second.

The question to investigate is how a given implementation of DRP can be optimized to meet such requirements. Is this even possible? If not, where do the main limitations come from? How can we optimize our design to overcome such limitations? In our opinion, the main trade-off comes from the decoupling between the various components. It brings flexibility and mitigates interference, but at the cost of a larger minimum end-to-end deadline that can be supported.

IV. Conclusions

Concealing hard real-time guarantees and wireless communication is very challenging. However, the emergence of Glossy-based protocols help significantly, as they eliminate the need for complex routing and enable unparalleled flexibility and adaptability in low-power wireless communications. Hence, we are striving to bridge this gap and eventually enable the successful deployment of safe and reliable wireless CPS.

Acknowledgments. This work was supported by Nano-Tera.ch, with Swiss Confederation financing, and by the German Research Foundation (DFG) within the Cluster of Excellence "Center for Advancing Electronics Dresden" (CFAED).

References

[1] J. Åkerberg, M. Gidlund, and M. Björkman, "Future research challenges in wireless sensor and actuator networks targeting industrial automation," in *Proc. of IEEE INDIN*, 2011.

[2] F. Sutton, M. Zimmerling, R. Da Forno, R. Lim, T. Gsell, G. Giannopoulou, F. Ferrari, J. Beutel, and L. Thiele, "Bolt: A stateful processor interconnect," in *Proc. of ACM SenSys*, 2015.

[3] M. Zimmerling, L. Mottola, P. Kumar, F. Ferrari, and L. Thiele, "Adaptive real-time communication for wireless cyber-physical systems," ETH Zurich, Tech. Rep., 2016.

[4] F. Ferrari, M. Zimmerling, L. Mottola, and L. Thiele, "Low-power wireless bus," in *Proc. of ACM SenSys*, 2012.

[5] F. Ferrari, M. Zimmerling, L. Thiele, and O. Saukh, "Efficient network flooding and time synchronization with Glossy," in *Proc. of ACM/IEEE IPSN*, 2011.

[6] R. Lim, F. Ferrari, M. Zimmerling, C. Walser, P. Sommer, and J. Beutel, "Flocklab: A testbed for distributed, synchronized tracing and profiling of wireless embedded systems," in *Proc. of ACM/IEEE IPSN*, 2013.

[7] M. Zimmerling, F. Ferrari, L. Mottola, and L. Thiele, "On modeling low-power wireless protocols based on synchronous packet transmissions," in *Proc. of IEEE MASCOTS*, 2013.