IX
Conclusion

We presented Céu, a system-level programming language targeting control-intensive WSN applications. Céu is based on a synchronous core that combines parallel compositions with standard imperative primitives, such as sequences, loops and assignments. Our work has three main contributions:

– A resource-efficient synchronous language that can express control specifications concisely.

– The stack-based execution policy for internal events as a powerful broadcast communication mechanism.

– A wide set of compile-time safety guarantees for concurrent programs that are still allowed to share memory and access the underlying platform in "raw *C*".

We argue that the dictated safety mindset of our design does not lead to a tedious and bureaucratic programming experience. In fact, the proposed safety analysis actually depends on control information that can only be inferred based on high-level control-flow mechanisms (which results in more compact implementations). Furthermore, Céu embraces practical aspects for the context of WSNs, providing seamless integration with *C* and a convenient syntax for timers.

As far as we know, Céu is the first language with stack-based internal events, which allows to build rich control mechanisms on top of it, such as a limited form of subroutines and exception handling. In particular, Céu's subroutines compose well with the other control primitives and are safe, with guaranteed bounded execution and memory consumption.

We presented two complete demos to show how typical patterns in WSNs such as sampling, timeout and concurrency can be easily implemented. They also explore parallel compositions for specifying complementary activities in separate. Communication among activities can either use internal events or safe access to global variables.

Our evaluation compares several implementations of widely adopted WSN protocols in Céu to *nesC*, showing a considerable reduction in code size with a small increase in resource usage. On the way to a more in-depth qualitative approach, such as evaluating the leraning curve of the language, we have been teaching CÉU as an alternative to *nesC* in hands-on WSN courses in a high school for the past two years (and also in two universities in short courses). Our experience shows that students are capable of implementing an *SRP*-like routing protocol in CÉU in a couple of weeks.

We presented a formal semantics for the control aspects of CÉU and discussed how they are implemented in *C*. The resource-efficient implementation of CÉU is suitable for constrained sensor nodes and imposes a small memory overhead in comparison to handcrafted event-driven code.

We believe that the strong position in favor of shared-memory concurrency is also a contribution of the thesis: first because although synchronous languages emerged in the early eighties, we are not aware of derived work allegedly addressing this issue; second because the current trend in the programming-languages community is towards the adoption of more pure functional languages and message-passing concurrency to get rid of shared memory, which is in the opposite direction of CÉU.