Demo Abstract: TinyOS 2.0

The TinyOS 2.x Working Group

http://www.tinyos.net/scoop/special/working_group_tinyos_2-0

Categories and Subject Descriptors

D.4.0 [**Operating Systems**]: General

General Terms

Design, Documentation, Reliability, Standardization

A sensor network operating system has to be efficient, flexible, and reliable. Energy contraints mean that nodes not only have limited resources, but they must use those resources sparingly. As deployments are application domain specific, a flexible OS lets software make the most of these limited resources in order to meet that applications requirements without inefficiencies introduced by generality. Finally, in order for deployments to last unattended for long periods of time, the underlying OS must be stable and reliable.

The past five years have seen tremendous research and technological developments in wireless sensor networks. The TinyOS operating system has been able to adapt and grow with this progress. However, five years of experience have shown that some of its structure and abstractions to be problematic. The huge existing TinyOS codebase means that changing basic aspects of the OS would break all of this existing work.

This has led us to develop TinyOS 2.0, a second-generation mote operating system. TinyOS 2.0 keeps many of the basic ideas in TinyOS 1.x: it is a component-based operating system written in the nesC language [1]. However, it is a clean-slate rewrite of the entire OS. Every major subystem has a corresponding TEP (TinyOS Enhancement Proposal) document, which describes the design and structure of the subsystem and documents a sample implementation. TEPs are open to review and comment from the larger community. As having a broad base of experience is important to guide design and implementations, the TinyOS 2.0 Working Group (WG) includes developers from UC Berkeley, TU Berlin, Stanford, UCLA, Crossbow, Moteiv, Arched Rock and Intel.

TinyOS 2.0 pushes forward in three key areas: greater platform flexibility, improved robustness and reliability, and the concept of service distributions. The first is intended to simplify platform development and interoperability, the second is a general design goal, and the last simplifies application development.

TinyOS 2.0 supports greater platform flexibility in two ways. First, it introduces a three-layer *Hardware Abstraction Architecture* (HAA) [2]. The bottom layer is the Hardware Presentation Layer (HPL), which provides access to basic resources, such as registers, interrupts, and pins, through nesC interfaces. The middle layer is the Hardware Abstraction Layer (HAL), which has higher-level interfaces that provide useful abstractions of the full capabilities of the underlying hardware. The top layer is the Hardware Independent Layer (HIL), which presents abstractions that are hardware independent and therefore cross-platform.

Second, TinyOS 2.0 uses version 1.2 of the nesC language, which has new features to better support cross-platform networking. nesC 1.2 introduces the notion of a *network type* at a language level: programs can declare structs and primitive types that follow a cross-platform (1-byte aligned, big-endian) layout and encoding. This allows services to declare cross-platform packet formats without resorting to macros or explicit marshaling/unmarshaling. Experimental data shows that the overhead of this encoding is negligible.

TinyOS 2.0 improves system reliability and robustness by redefining some of the basic TinyOS abstraction and policies, such as initialization, the task queue, and power management. For example, in TinyOS 1.x, all components share a fixed-size task queue, and a given task can be posted multiple times [3]. This causes a wide range of robustness problems, as if a component is unable to post a task due the queue being full, it may cause the system to hang. In TinyOS 2.0, every task has its own reserved slot in the queue, and can be only posted once. Our initial experiences have shown these semantics to greatly simplify code and lead to more robust components.

Finally, TinyOS 2.0 introduces the notion of *service distributions*. A service distribution is a collection of services whose underlying components have been designed and composed to work well together. A service distribution is essentially a system API. Because the services are designed to work together as a whole, users do not have to worry about initialization order, configuration, or the details of the underlying implementations. Service distributions also introduce power management policies on top of the basic OS mechanisms. As a whole, they introduce a seperation of concerns between low-level systems and applications, simplifying application development. TinyOS 2.0 supports service distributions by defining simple OS mechansims and interfaces that can be reused in many fashions, and through new nesC features: a service is a component which an application can instantiate.

TinyOS 2.0 has been in development for just over a year, and each of its subsystems has been revisited and redesigned several times. Its first prerelease to the community was at the end of August, 2005, and the first full release is slated for the end of September. The inception of the TinyOS 2.0 working group and its success so far has led members of the TinyOS community to start a few other, smaller WGs. Going forward, one of the greatest challenges in these efforts will be fostering collaboration and managing the creation and agendas of these working groups in order to push sensor networks further towards ubiquity and deployment.

REFERENCES

[1] D. Gay, P. Levis, R. von Behren, M. Welsh, E. Brewer, and D. Culler. The nesC language: A holistic approach to networked embedded systems. In *SIGPLAN Conference on Programming Language Design and Implementation (PLDI'03)*, June 2003.

[2] V. Handziski, J. Polastre, J.-H. Hauer, C. Sharp, A. Wolisz, and D. Culler. Flexible hardware abstraction for wireless sensor networks. In *Proceedings of the Second European Workshop on Wireless Sensor Networks (EWSN)*, Feb. 2005.

[3] P. Levis, S. Madden, J. Polastre, R. Szewczyk, K. Whitehouse, A. Woo, D. Gay, J. Hill, M. Welsh, E. Brewer, and D. Culler. TinyOS: An operating system for wireless sensor networks. In *Ambient Intelligence*, New York, NY, 2004. Springer-Verlag.