

TOSThreads: Thread-safe and Non-Invasive Preemption in TinyOS

Kevin Klues[‡], Chieh-Jan Mike Liang[†], Jeongyeup Paek[°], Răzvan Musăloiu-E.[†], Philip Levis^{*}, Andreas Terzis[†], Ramesh Govindan[°]

TOSThreads: Fully preemptive threads library with efficiency of event-based kernel

Goal

- **Thread-Safe, Fully preemptive application-level threads**
 - Enable higher-level application development without needing to explicitly manage yield or continuations, or partition a long-running computation
- **Non-Invasive, Efficiency of event-based kernel**
 - Timing-sensitive aspects of TinyOS must be preserved
- **Backward compatible with existing TinyOS code**
- **Supports evolvable thread-safe kernel**
- **Enable flexible application development**
 - Through dynamic linking and loading

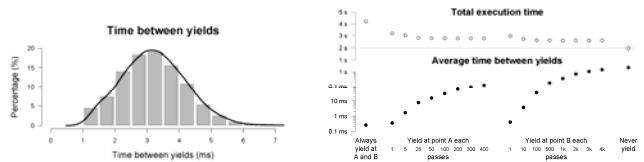
Approach

- **Applications run inside application threads**
 - Only execute whenever the kernel thread becomes idle.
- **Single higher priority TinyOS kernel thread**
 - Preserve all of the efficiency and timing constraints of TinyOS.
- **Message Passing**
 - Application threads invoke kernel operations via message passing
- **TinyLD**
 - Leveraging the user/kernel boundary, dynamically links applications to a static kernel

Challenges: Preemption is difficult on resource constrained nodes

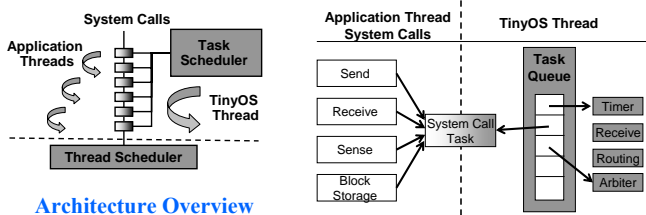
Kernel must be reentrant, thread-safe, and non-invasive

- **Thread preemption should not cause the kernel to fail**
- **Cooperative threading is not easy**
 - Rely on applications to explicitly yield the processor and manage concurrency.
 - Determining the right strategy for inserting yield points in a long running computation is a non-trivial task
- **Kernel locking is inefficient**
 - There is a basic trade-off between locking overhead / complexity and how much

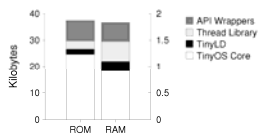


Implementation and Evaluation: Improved usability and expressivity with little overhead

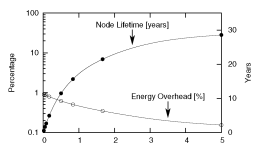
TOSThreads



Architecture Overview



Memory footprint



Energy overhead

Kernel APIs are blocking system calls wrapped around event-driven TinyOS services

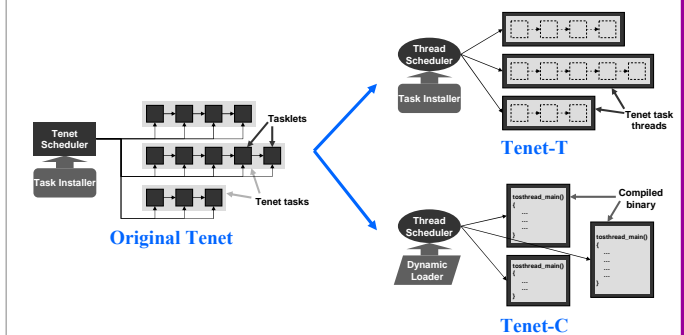
TOSThreads context switches and system calls introduce an overhead of less than 0.92%.

TinyLD requires less than 90ms on a representative sensing application

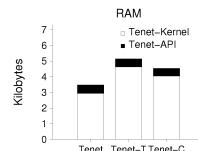
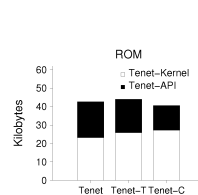
TOSThreads has been used in various projects:

- Latte, Johns Hopkins University
- MAMMARK, University of California, Santa Cruz
- SPINE, Telecom Italia
- Tenet, USC

Tenet API on top of TOSThreads

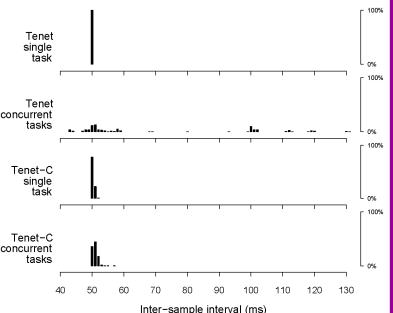


Original Tenet



Improved expressivity with low memory overhead

• TaskLong : Repeat(100ms) → ReadBlock(A,1024) → Avg(B,A) → MeanDev(C,A) → Send()
 • TaskSample : Repeat(50ms) → TimeStamp(A) → Sample(ADC5,V) → Send()



Preemption is non-invasive, and improves Tenet concurrency