SYS1: Avrora: Scalable Sensor Network Simulation with Precise Timing

Authors
Ben Titzer
Daniel K. Lee
Jens Palsberg

Publication Date
2005

Center for Embedded Networked Sensing

# Avrora: Scalable Simulation of Sensor Networks with Precise Timing

**Ben L. Titzer** and **Jens Palsberg**

UCLA Compilers Group - http://compilers.cs.ucla.edu/

## Introduction: Sensor networks are hard to develop and test

### Debugging microcontroller programs is hard

- **Narrow debugging interface to hardware**
  - Interface to chip is ***narrow*** and does not allow complex interaction with the executing program in real time
  - presence of debugging code influences results
- **Intricate, low-level code**
  - Driver code for various ***hardware sensors*** and communications devices
- **Subtle timing interactions**
- **Longer development cycles due to reprogramming**

### Distributed network behavior

- **Behavior of code depends on environment**
  - Can depend on ***input*** to sensors and ***communication*** with outside world
- **Distributed, multi-hop communications**
  - Routing algorithms
  - Data mining
- **Detailed network monitoring difficult**
  - Lots of sensors, lots of communication
  - Microsecond level phenomenon
  - Complex interactions

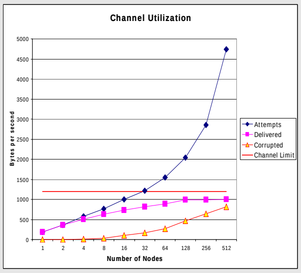
## Problem Description: Accurate simulation of sensor programs requires precise timing

### High Accuracy needed

- **Cycle-level phenomenon**
  - Software control of radio hardware device
  - Sleep behavior, interrupt behavior
  - Measure time-dependent quantities such as channel utilization, access latency
- **Previous approaches don’t scale well**
  - Synchronization of nodes every clock cycle
  - Each device simulated adds work every clock cycle
  - Poor performance for large networks

### Core Problems Identified

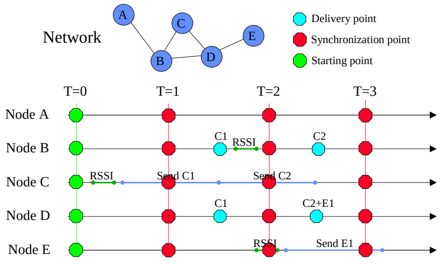
- **Send-Receive Problem**
  - A receiver node must not proceed past a point in time at which it should receive a radio packet from a sender until the sender is beyond the point of transmission
  - Time to send a byte on mica2: ***6106*** clock cycles
- **Sampling Problem**
  - A node sampling the RSSI value of its own radio should not proceed past a point in time at which possible senders can influence the sampled value
  - Time to sample RSSI on mica2: ***832*** clock cycles



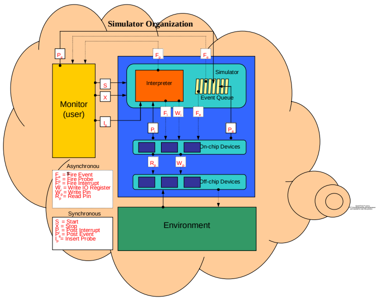
## Proposed Solution: Efficient, parallel, cycle-accurate simulation with Avrora

### Parallel machine-code simulation

- **Map one thread per node**
  - Allows ***parallelism*** in simulation
  - Requires new solutions to ***Send-Receive*** and ***Sampling*** problems
- **Synchronization Interval approach**
  - Periodically synchronize threads to preserve ***order*** and ***timing*** of communications
- **Wait For Neighbors approach**
  - Node waits for all neighbors that can influence its operation to pass a specific point in global simulation time



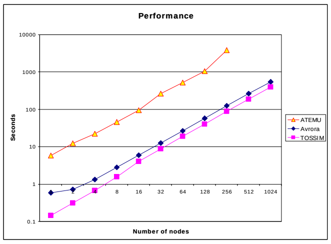
Synchronization Illustration

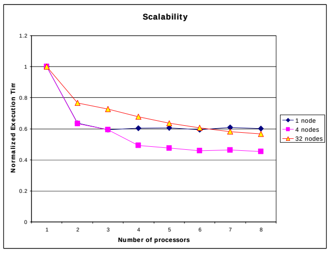


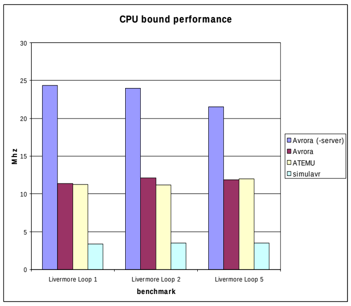
### Results

- **Cycle-accurate AVR simulator**
  - Efficient execution of program code
  - Accurate timing of program interaction with devices
- **Device and Radio simulation**
  - Timers, UART, SPI, CC1000
  - Important for correct program simulation
- **Whole network simulation**
  - Nodes sense, compute, and communicate, with full monitoring capabilities
- **Profiling and Monitoring**
  - Flexible extension points allow for detailed monitoring of program execution without changes to simulator

**Avrora allows sophisticated program profiling to be performed during simulation without loss of precision, and with fully cycle-accurate results.**







Simulator comparison

| | Portability | Scalability | Parallelism | Performance | Accuracy | Flexibility | Program Analysis | Requires |
|---|---|---|---|---|---|---|---|---|
| Avrora | Excellent | Good | Good | High | High | Excellent | Framework<br>Stack<br>CFG | Java |
| TOSSIM | Poor | High | None | Excellent | Fair | Poor | none | Linux<br>X86<br>TinyOS<br>ncc<br>gcc |
| AtEmu | Fair | Poor | None | Fair | Excellent | Good | none | GTK2<br>X<br>Libelf<br>gcc<br>libxml |

UCLA – UCR – Caltech – USC – CSU – JPL – UC Merced