

Performance Aware Tasking for Environmentally Powered Sensor Networks

Aman Kansal, Dunny Potter, Mani Srivastava

**Networked and Embedded Systems Laboratory,
EE Dept., University of California, Los Angeles**

Acknowledgements:



What are sensor networks?

- **Networks of numerous deeply embedded devices which sense (and control) the environment**
 - Wireless, low energy, low datarate, autonomous



UC Berkeley's deployment at Great Duck Island



At Huntington Botanical Gardens



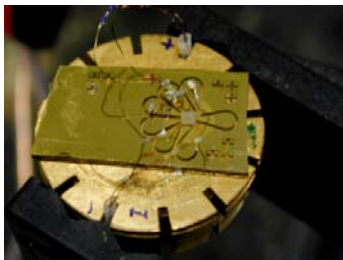
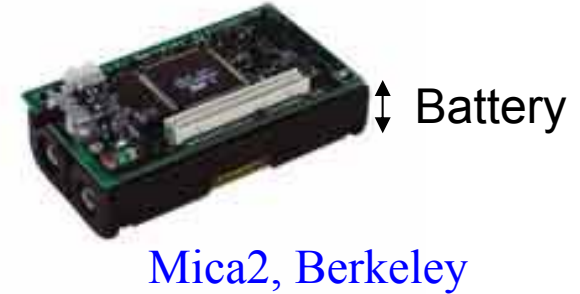
UCLA's seismic array (under construction)



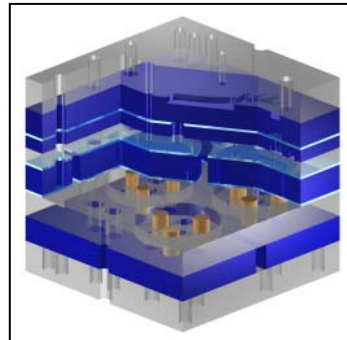
NESL-UCLA's localization array

Energy Harvesting

- Batteries are too big
- Batteries don't last forever
- Methods exist to extract energy from the environment



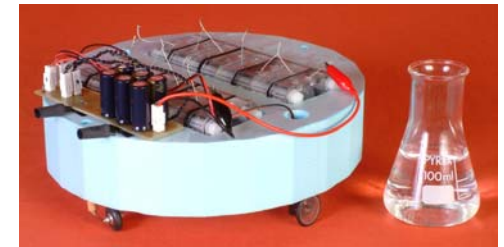
Thermoelectric
(DARPA, JPL,
CalTech)



Micro-Hydraulic
Transducer
(DARPA, MIT)



Solar Cells



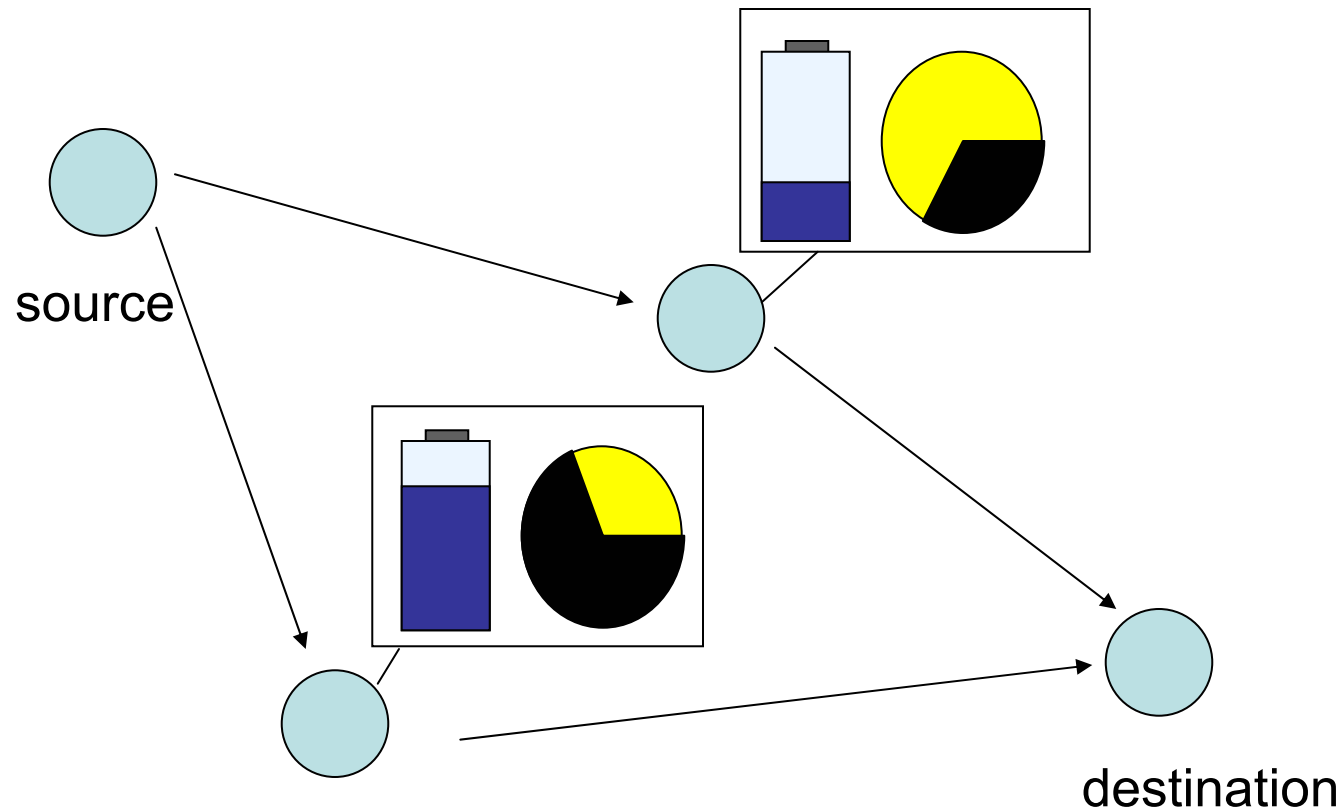
Bio-Fuel, U Bristol

Managing Harvested Energy

- **Its different from battery energy**
 - Supply varies with time
 - Need to adapt performance
 - Supply varies in space
 - Different nodes get different energy: need load sharing
 - Supply is repetitive (does not die out)
 - Opportunity to last forever
 - Efficiency concerns
 - Match load to maximize transfer
 - Supply direct when possible, instead of through battery

Example

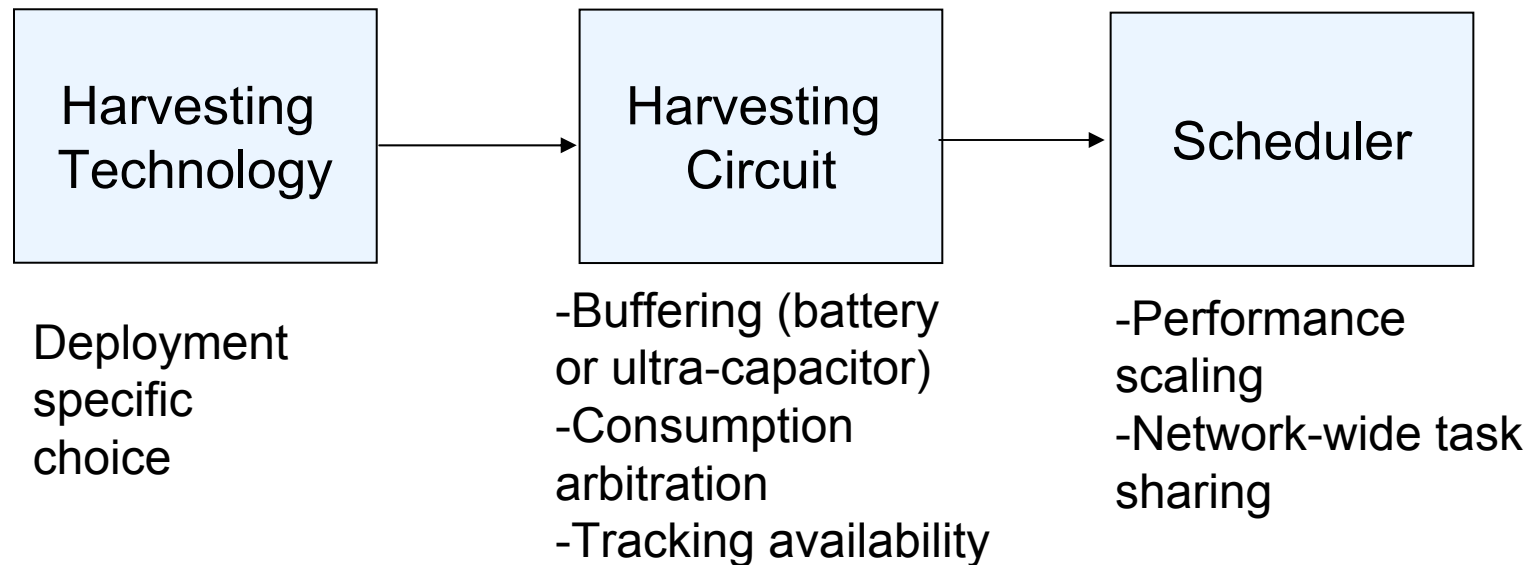
Which route to choose?



Managing Harvested Energy

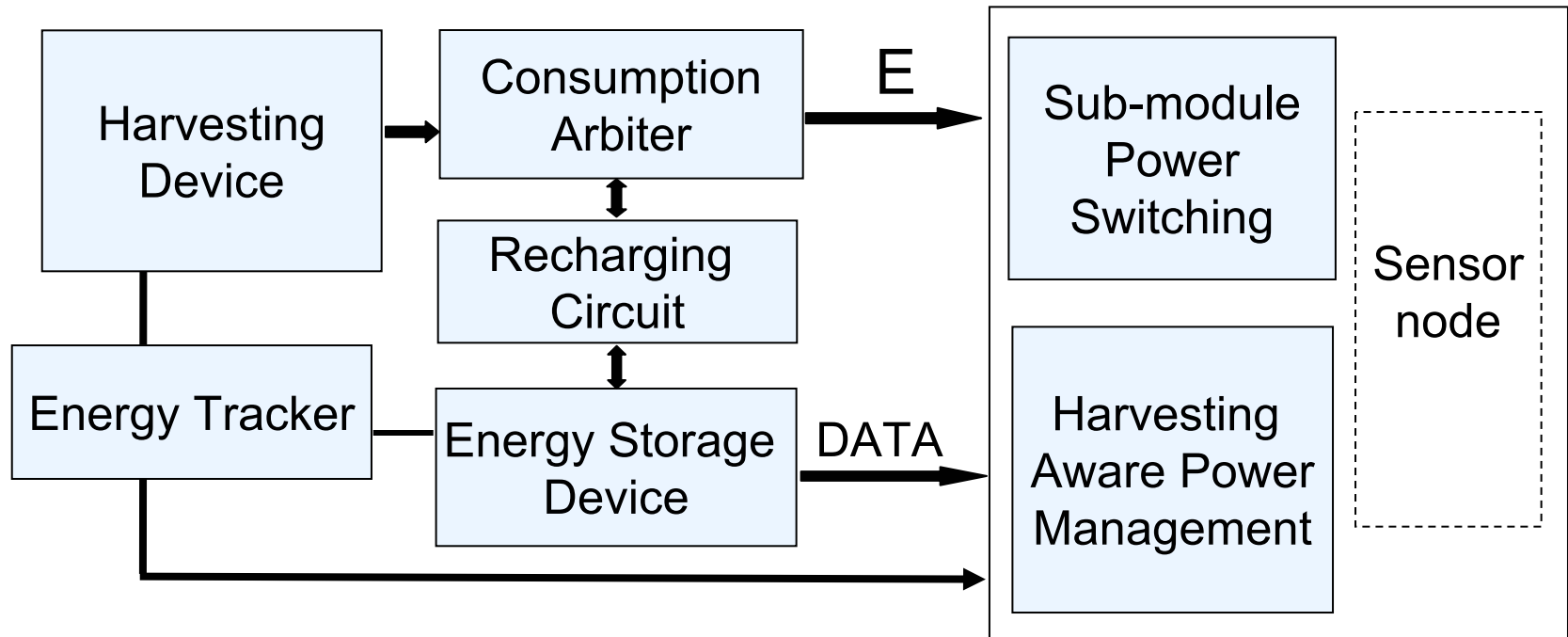
- **Key issues:**

- Use the available energy most efficiently
- Estimate achievable performance level



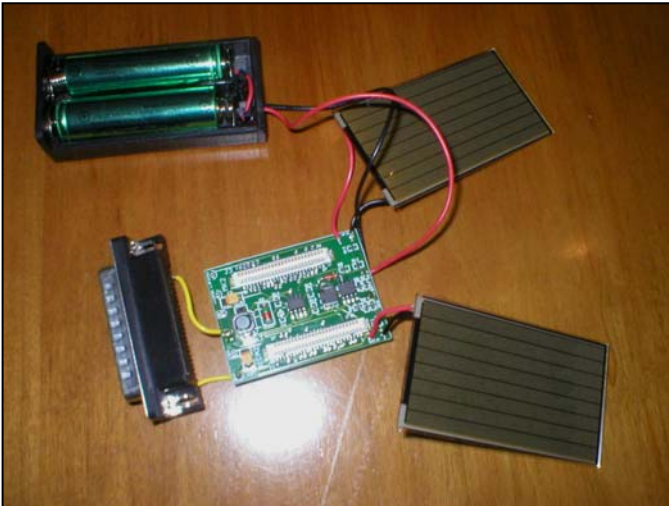
Harvesting Circuit

- **System Block Diagram**

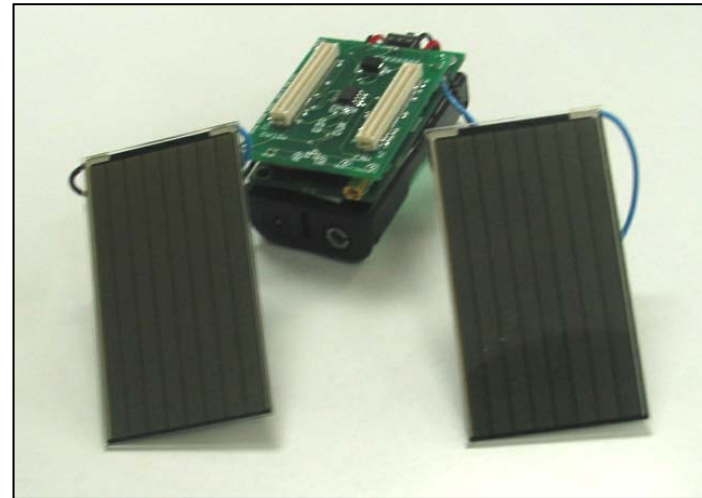


Implementation

- **TinyOS Device Drivers for new harvester hardware provide simple user interface:**
 - `async command result_t getData()`
 - `async event result_t dataReady`



Harvesting Circuit



Heliomote

Scheduler Design

Problem

Environmental energy supply is variable

Supply is independent of demand

Existing Approach

Recharge battery and let the load use energy as required

Battery makes up for discrepancy, node dies when out of battery

Our Approach

Track environmental availability

Scale performance: match availability, last forever

Source Characterization

What is available from the variable environmental source?

- **Leaky bucket like model:**

- Leaky bucket models bursty Internet traffic
 - We are modeling energy supply
- So, add a constraint for minimum flow

- **Definition: $E(t)$ is a $(\rho, \sigma_1, \sigma_2)$ source if for all T :**

$$\int_0^T E(t) \geq \rho T - \sigma_1 \quad \text{and} \quad \int_0^T E(t) \leq \rho T + \sigma_2$$

Questions of interest

Given the source parameters:

- **What's achievable application throughput or latency?**
- **Can the system last eternally at required performance level?**
 - What additional resources are required, if not?
- **Considering efficiencies of battery and various power modes how should the tasks be scheduled?**

Harvesting Theory

THEOREM 1: If

- a system is powered by a $(\rho, \sigma_1, \sigma_2)$ source
- has energy storage capacity $\geq (\sigma_1 + \sigma_2)$
- operates at constant power level ρ

then

- 1. it utilizes the energy source fully**
- 2. can survive forever**

- Larger battery gives no long term performance gain
- At least a rate of energy consumption = ρ can be supported

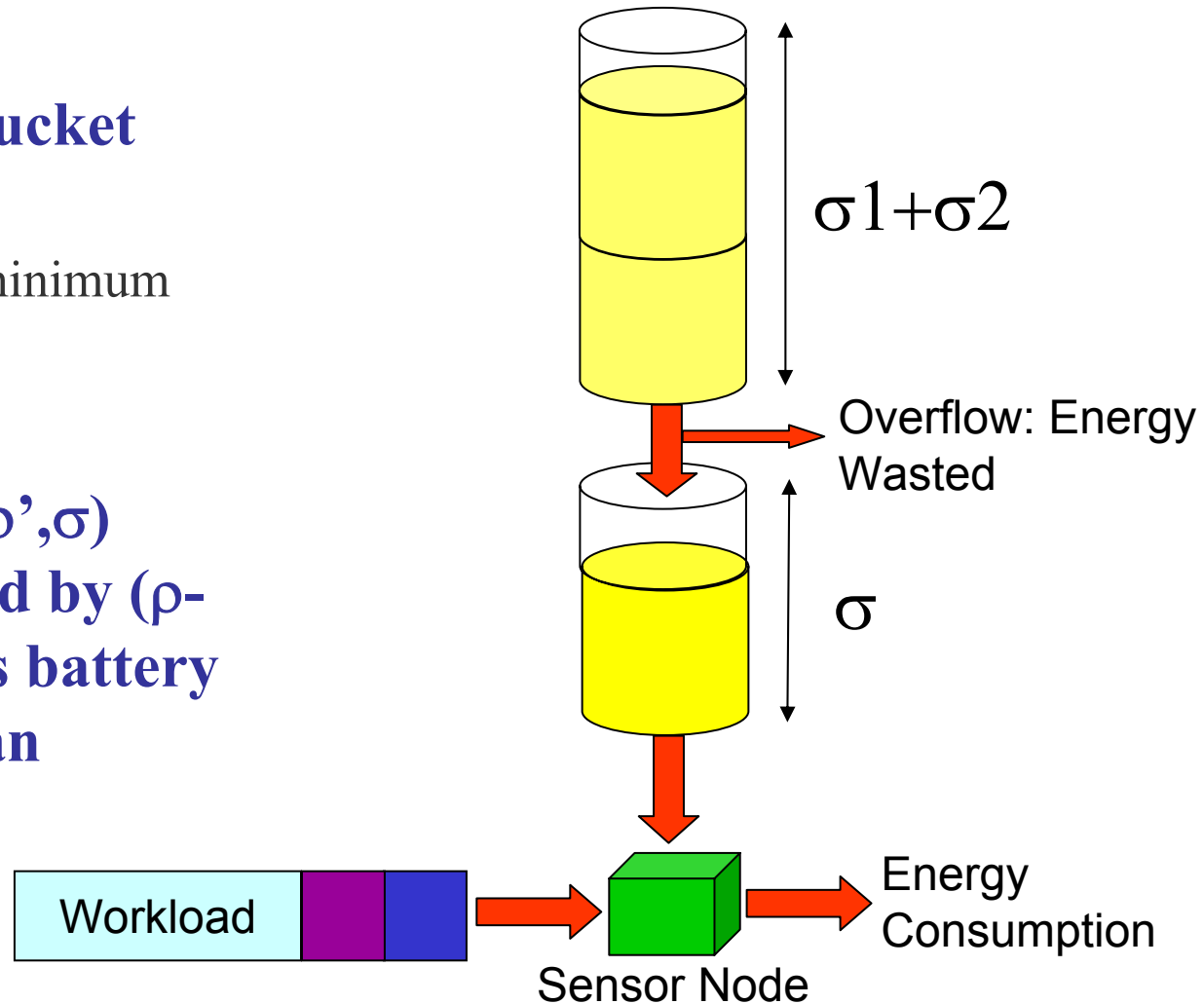
Proof Outline

Proof (by contradiction)

- Maximum energy we can use = everything provided by environment
 - Maximum ρ = average rate of energy
- No energy from the environment is wasted: ensured by first condition
 - Assume Initialization phase during which battery charged to σ_2
 - Consider the first instant when some small energy ΔE overflows (in time Δt), then $E(\Delta t) = \rho\Delta t + \Delta E$
 - Consider the preceding time duration t in which the battery had filled up σ_1 capacity: $E(t) = \rho t + \sigma_1$
 - Then $E(t+\Delta t) = \rho t + \sigma_1 + \rho\Delta t + \Delta E > \rho(t + \Delta t) + \sigma_1$ which is a contradiction
- Similarly using other condition can prove that ρ can always be supported

Not Operating at Constant Power

- **Assume a leaky bucket consumer (ρ', σ)**
 - No constraint on minimum energy usage
- **Theorem 2: If a (ρ', σ) consumer powered by $(\rho - \sigma_1 - \sigma_2)$ source, has battery size $\sigma + \sigma_1 + \sigma_2$, it can survive eternally.**



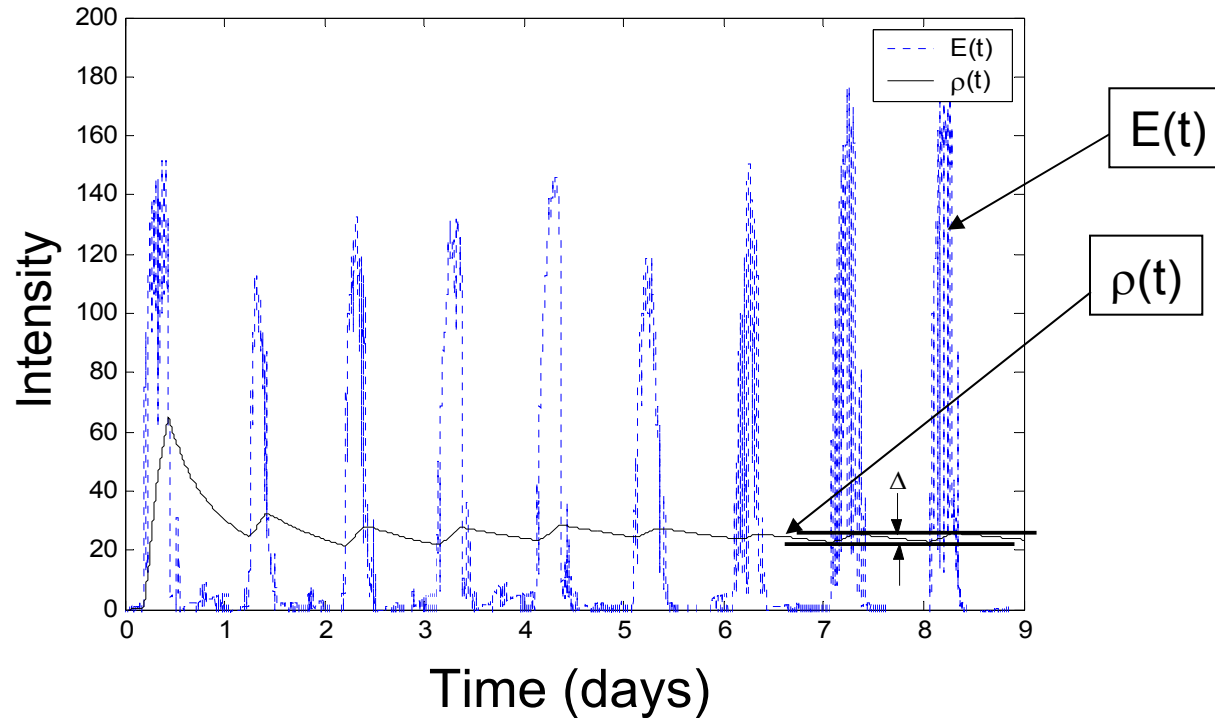
Single Server Harvesting

- **Need to determine $\sigma_1, \sigma_2, \rho, \rho', \sigma$ from measured energy availability and load**
 - Can then provide estimate of feasible performance region
- **Source parameters**
 - Estimate ρ from a running average until difference between recent maxima and minima is within an acceptable margin
 - Works for a cyclic source (such as the sun)

$$\rho(t) = \frac{\int_0^t E(t) dt}{t}$$

- Can estimate σ_1, σ_2 by locating largest underflow and overflow
 - Done only at design time to estimate required battery size

Measured Data: Single Node



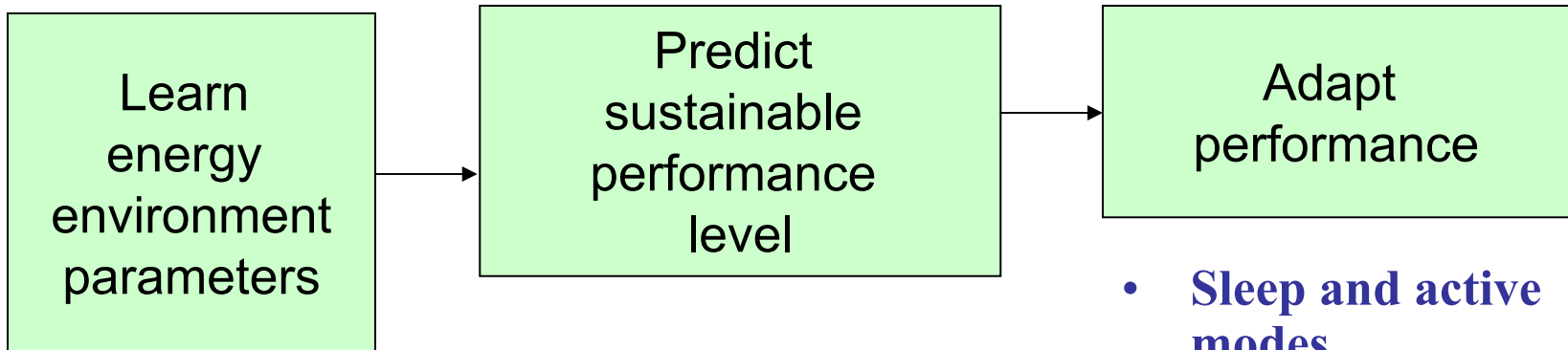
Using above data and Mica2 consumption characteristics:

$\rho = 23.6\text{mW}$ $\sigma_1 = 1.463\text{ MJ}$ $\sigma_2 = 1.856\text{ MJ}$ $\sigma = 153\text{ J}$

Theorem 2: battery required = $\sigma + \sigma_1 + \sigma_2 = 3.32\text{ MJ} = 922.43\text{mAh at } 3\text{V}$

Note: Larger battery will not help sustainable performance

Performance Control



$$\rho = xP_{\max} + (1-x)P_{\text{sleep}}$$

- **Sleep and active modes**
- **Dynamic Voltage Scaling**
- **Radio range control**
- **Sub-module power switching**

Multi-Server Harvesting

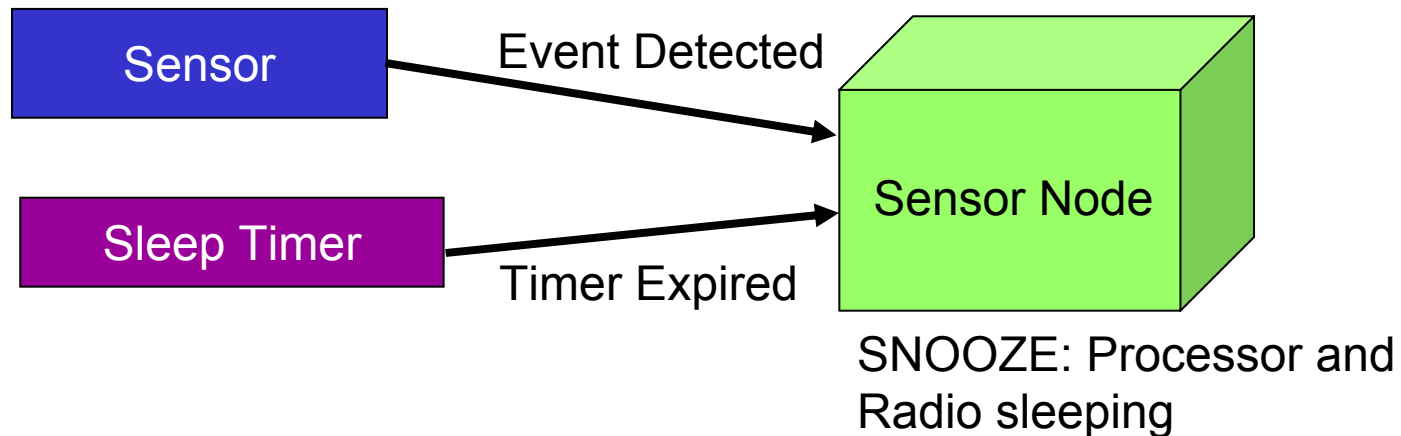
- **How can a distributed system manage the harvested energy to maximize performance of system as a whole**
 - energy resources vary across nodes,
 - task-load differs at different nodes,
 - some workload is share-able while some is not
- **Consider one energy intensive task: routing data**
 - Determine environmental energy aware communication strategy

Routing Options

- **Optimal routing is impractical**
 - Needs global traffic information for all times
- 1. **Nodes share state information and coordinate performance adaptation actions**
- 2. **Nodes adapt performance locally and routing protocol operates over sleepy nodes**
 - Cross-layer interdependence is minimized
 - Saves communication energy for coordination
 - Energy resource is known locally

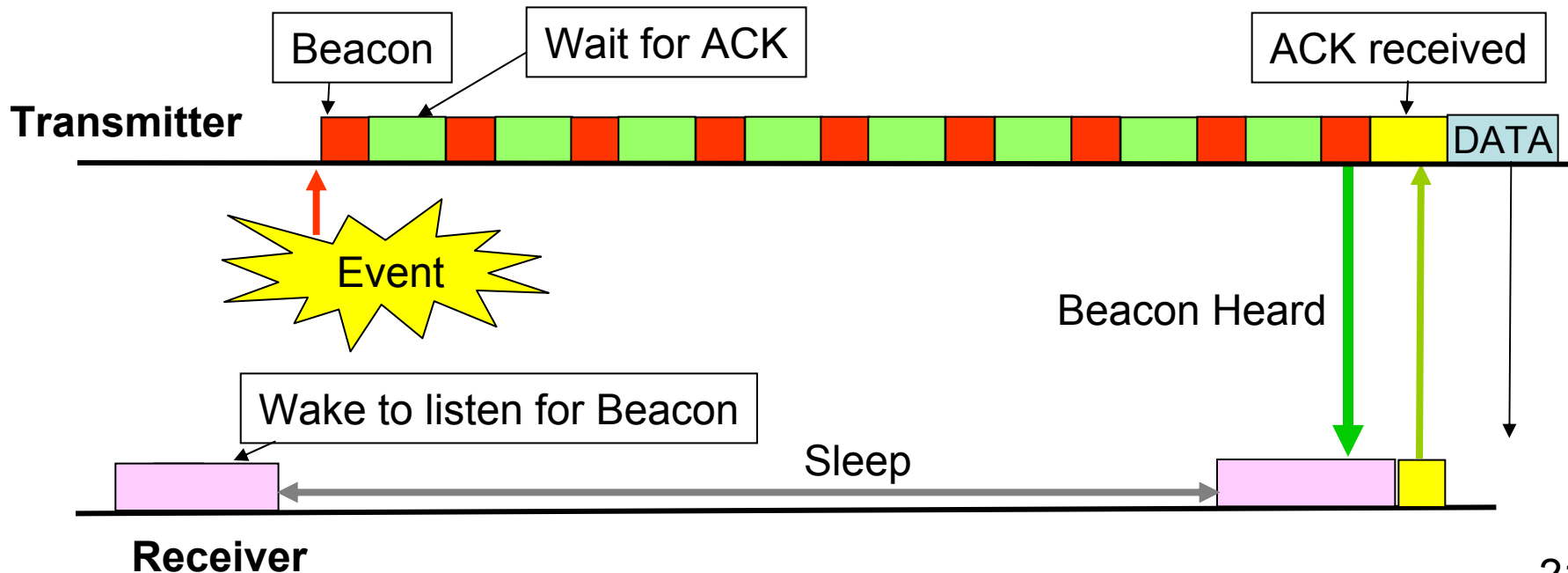
Practical Networking Method

- **Routing for an event monitoring sensor network**
 - Single sink (base station), multiple sources (nodes monitoring events)
 - Must report event when it occurs, otherwise no data
- **Measure energy and calculate duty-cycle locally**
 - Duty cycle determines latency of data relaying

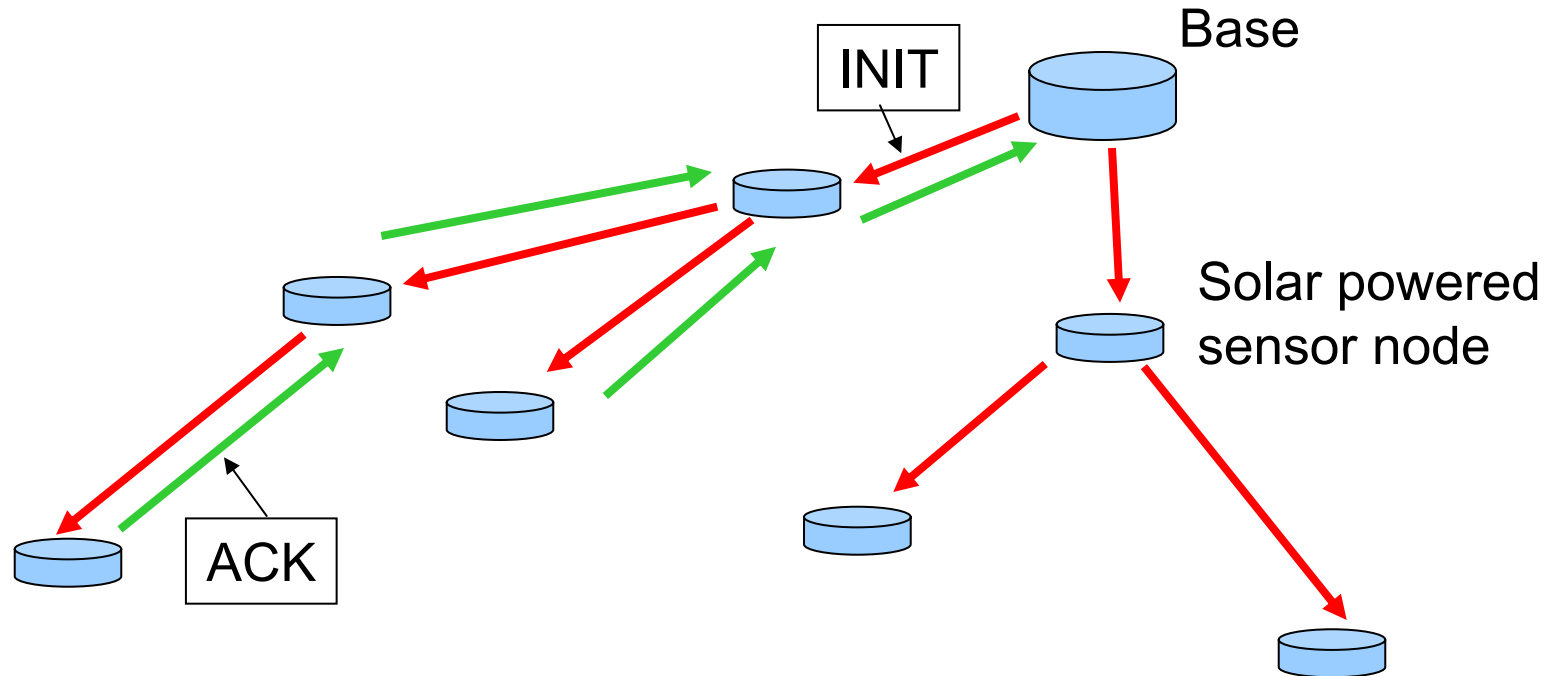


Communication with Sleep Mode

- Node can wake up if it has data to send
- How does a sleeping node receive data?
 - No time synchronization required between nodes



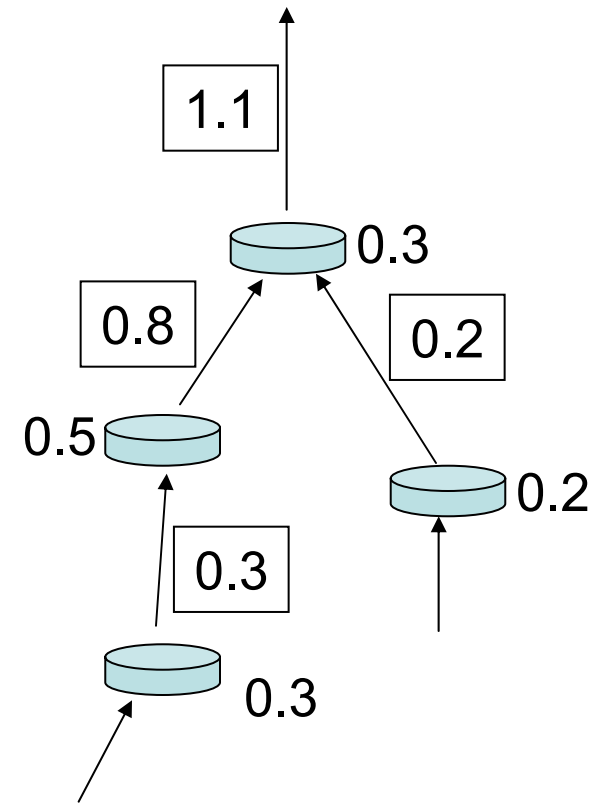
Routing Tree



- **Protocol**
 - Base station sends INIT
 - Receiver sends ACK and forwards INIT
- **Reverse path set up to base-station**
 - Possibly shortest, but not necessarily lowest latency

Network-wide Performance

- **Estimate network-wide latency constraint with observed environmental resource**
- **Central control over network latency is impractical**
 - sending all latencies to base station reduces scalability
- **Use in-network processing to compute path latency**
 - Receive path latencies from children
 - Forward highest plus own latency



Pseudo-code

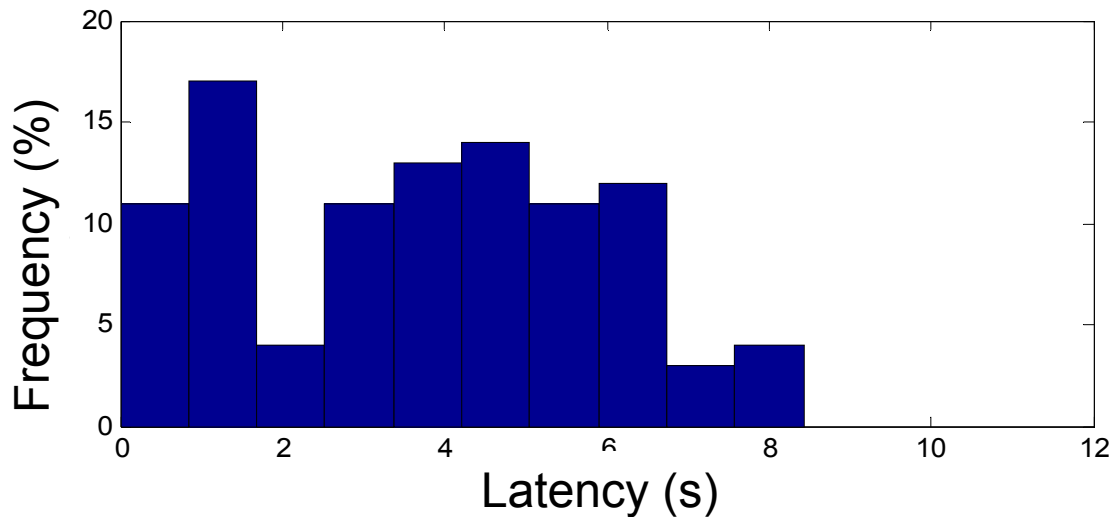
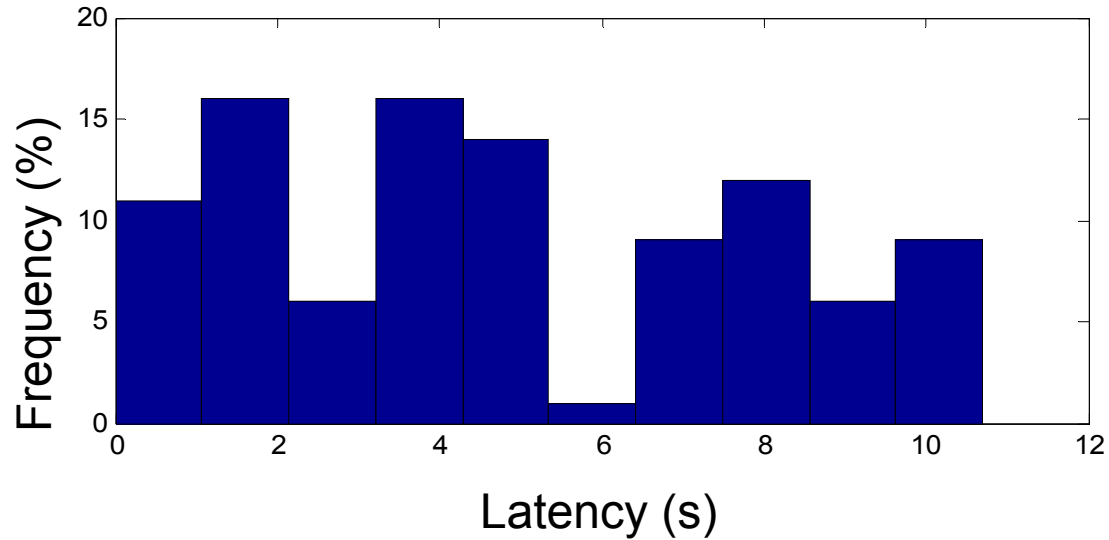
- **Initialize parameters:** $T_{\text{sleep}} = 0$, **childset = empty, childlatencylist = empty, parent = UNK, is_leaf = FALSE**
- **Spawn thread to estimate L. Generates $L_{\text{estimated}}$ event when finished.**
- **If received INIT message**
 - If parent == UNK
 - parent = sender-id from INIT message
 - Send ACK, Rebroadcast INIT message with own ID, Start ACK timer
- **If received ACK message**
 - childset = childset U sender-id from ACK
 - Delete ACK wait timer
- **If ACK wait timer expires, set is_leaf = TRUE**
- **If $L_{\text{estimated}}$ event received**
 - If is_leaf == TRUE
 - send LATENCY message to parent, Adjust T_{sleep} and duty cycle for this L
 - Else
 - If childlatencylist is complete
 - Calculate L and send LATENCY message
 - Adjust duty cycle and T_{sleep}
 - Else wait for childlatencylist to complete
- **If received LATENCY message store latency value in childlatencylist**

Simulation Study

- **Network with 50 nodes in random topology**
- **Solar data**
 - Real data was collected with a single node for 9 days.
 - Generated random data distributed uniformly around this, nodes assume to receive this energy
- **Latency performance of our protocol (fully distributed) compared to lowest latency route**
 - Lowest Latency Route: found using Bellman-Ford algorithm (link latency used as cost)
 - high message overhead

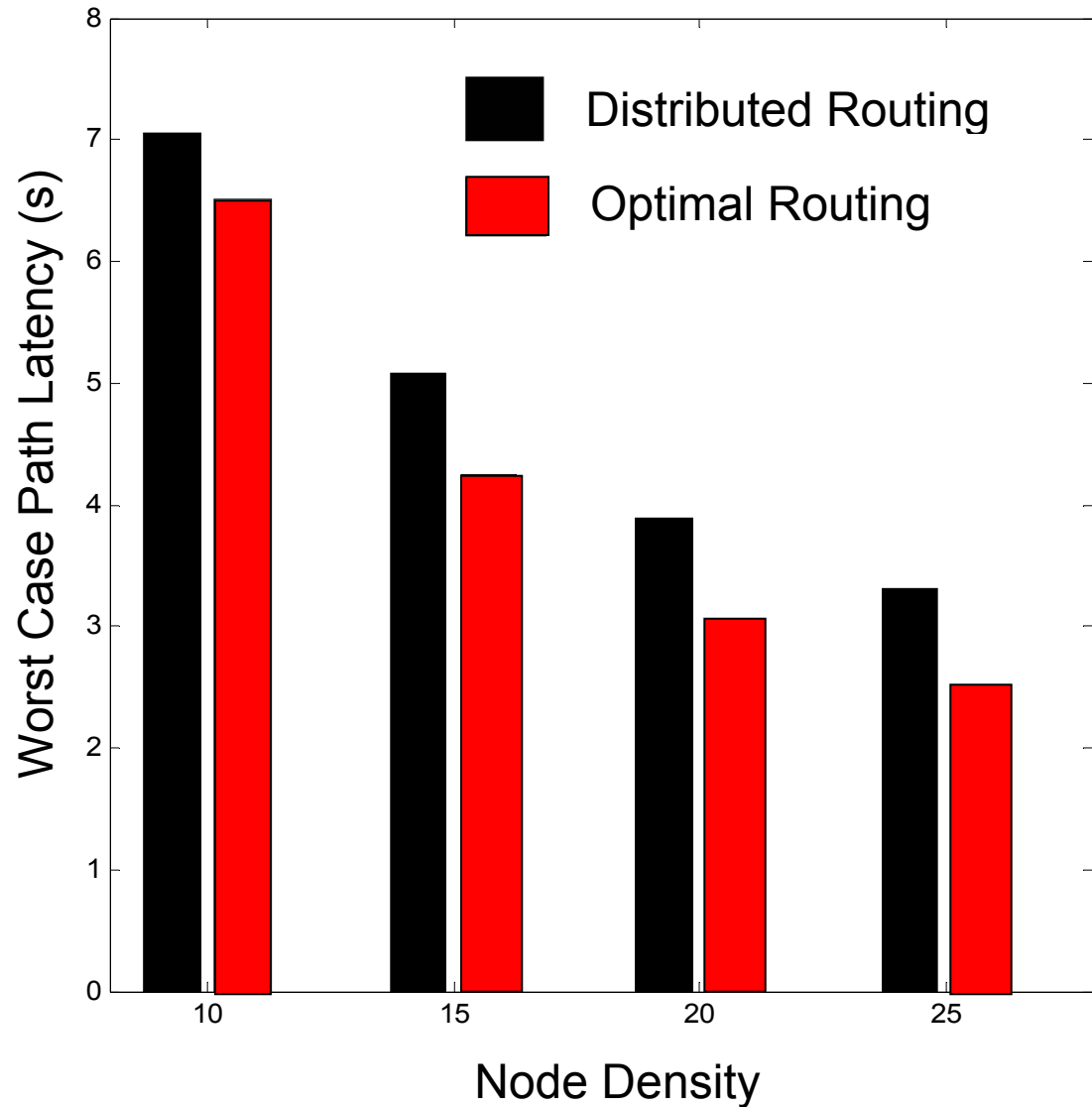
Simulation Study

Path Latency Histogram of a randomly deployed network



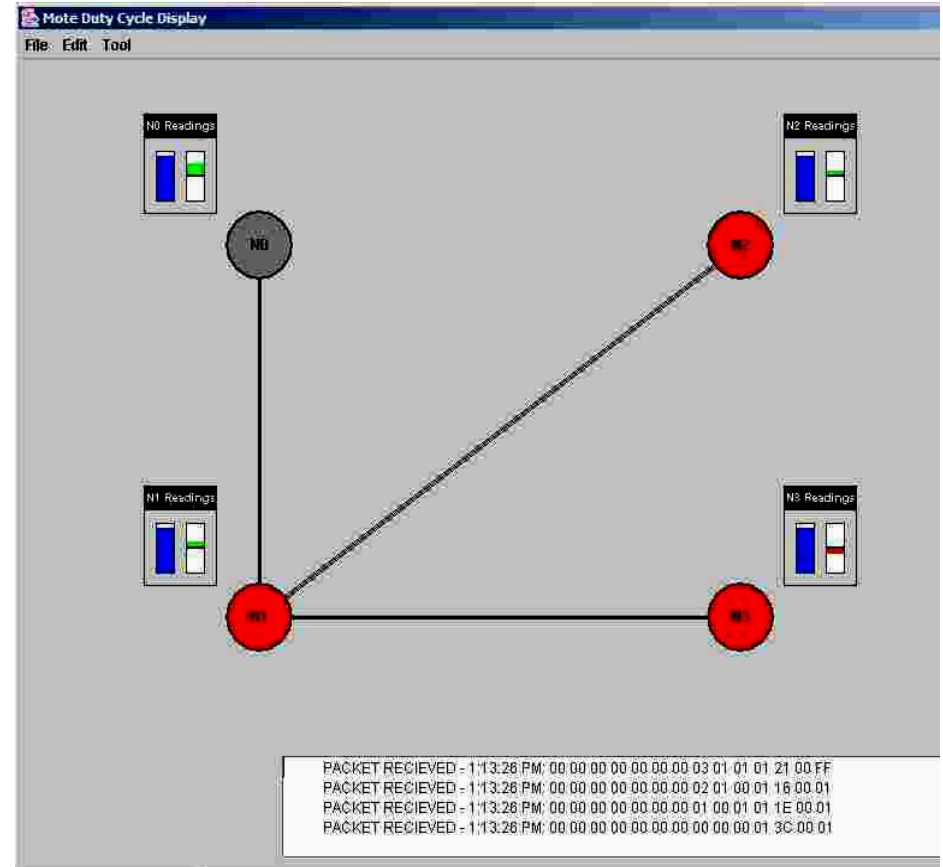
Simulation Study

Results averaged over 20 random topologies



Ongoing Work

- Test-bed to experiment with distributed harvesting-aware schedulers
- Apart from protocol messages, send each node characteristics to root node for monitoring
- Mote radio range larger than experiment space
 - Multi-hop created by address blocking



Conclusions

- **Harvesting technologies can enable long system lifetime**
 - Proof-of-concept system and algorithms to exploit environmental energy demonstrated
- **Methods are needed to measure and characterize harvesting sources**
 - Battery characterization is not sufficient
- **Distributed methods are required to optimally adapt global performance**
 - Schedule tasks appropriately in space and time to enhance performance