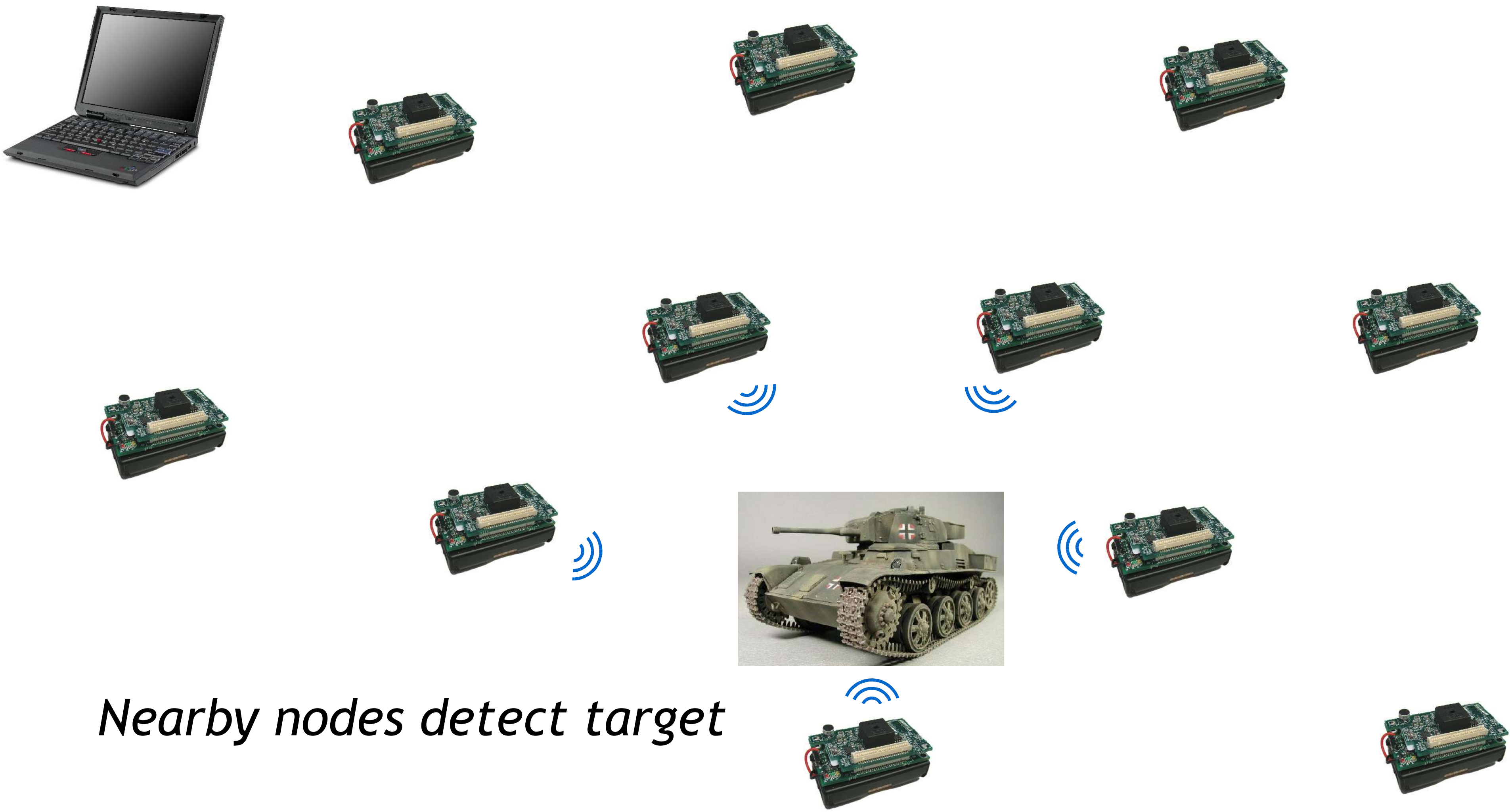
Decentralized, Adaptive Resource Allocation for Sensor Networks

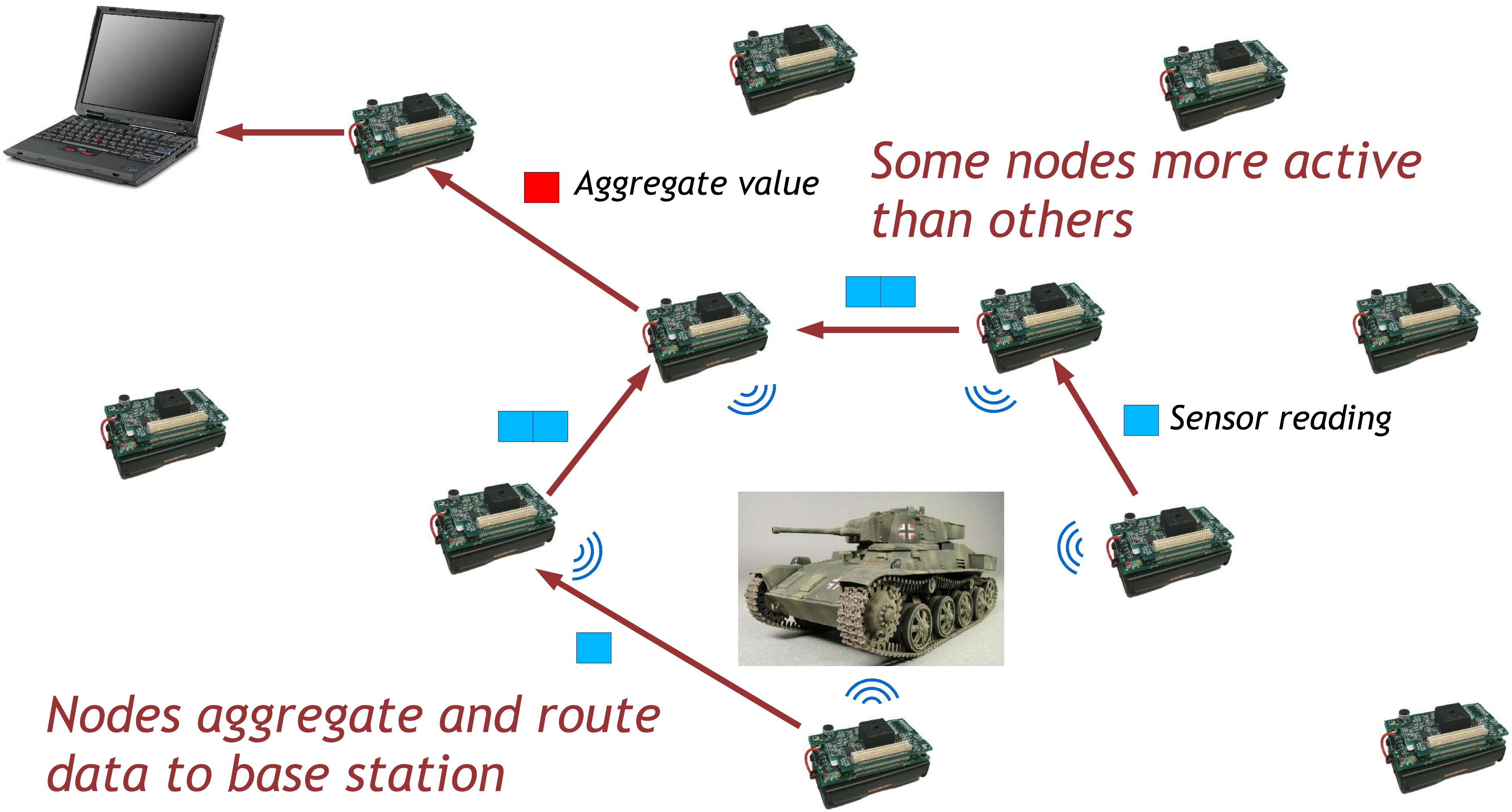
Geoff Mainland, David C. Parkes, and Matt Welsh
Division of Engineering and Applied Sciences
Harvard University

{mainland, parkes, mdw}@eecs.harvard.edu

Motivating example: Tracking



Motivating example: Tracking



Some nodes more active than others

Nodes aggregate and route data to base station

Problem Definition

The actions performed by each node have a deep impact on:

- **Accuracy and latency** of data returned to the base station
- **Radio contention**
- **Energy use** (and therefore, network lifetime)

Standard technique: *periodic duty cycling*

- e.g., "Every *T* seconds, sample the magnetometer, then transmit data if above threshold"
- Problem: Periodic sampling and communication not optimal for all nodes

Instead, nodes should *self-schedule* based on their local state

- Node should decide locally which operations to perform and how often
- Driven by interaction with environment

Node operation should *adapt* to changing conditions

- e.g., If interesting event happens nearby, node might ramp up sampling rate

Self-Organizing Resource Allocation (SORA)

SORA is an adaptive scheduling technique for sensor networks

- Rather than static scheduling, individual nodes tune their schedules over time
- No central control or dictated node program

Goal: Nodes should avoid *wasting energy*

- Every action taken by a node consumes some amount of enegy
- However, this energy is only *sometimes* "useful"

Example: Listening for incoming radio messages

- Consumes a lot of energy, but only useful if a message is actually received

Idea: Use feedback on which actions are useful to tune node behavior

- Nodes receive rewards when they take useful actions
- SORA uses reinforcement learning techniques to select best actions to take

Self-Organizing Resource Allocation (SORA)

Basic model:

- Nodes can select among a set of *actions*
- Each action has an associated *energy cost*
- When an action is "successful," the node earns a *reward*

Examples of actions (*energy cost measured on MicaZ motes*):

- *Sample* a sensor *(energy cost 84 μJ)*
- *Listen* for incoming radio messages *(energy cost 5.9 mJ)*
- *Transmit* a radio message *(energy cost 2.4 mJ)*
- *Aggregate* multiple sensor readings into a single value *(energy cost 84 μJ)*

Each node attempts to *maximize its reward*

- That is, subject to energy constraints
- We assume that nodes can determine *locally* which actions were useful

Utility Function

SORA drives action selection by assigning each action a *utility*

The utility for an action is a function of:

Reward for taking a successful action

- Advertised by base station and propagated to entire network
- Make use of lightweight data dissemination protocols (e.g., Trickle)

Energy availability

- Taking an action must stay within the node's energy budget
- We model nodes as having an *energy reserve* that is replenished at a constant rate

Data dependencies

- Cannot aggregate data until multiple samples have been received
- Cannot transmit if nothing in local buffer

Learning Expected Rewards

The utility function is the *expected reward* for taking an action:

$$u(a) = \begin{cases} \beta(a) \times reward(a) & \text{if the action's dependencies have been met} \\ 0 & \text{otherwise (e.g., not enough energy)} \end{cases}$$

$\beta(a)$ is the estimated *probability of success* for action a

- $\beta(a)$ is learned over time using an exponentially weighted moving average (EWMA)

When $u(a)$ for all actions is zero, node performs the “sleep” action

- Places node in lowest-power state for a short period of time (0.25 sec)

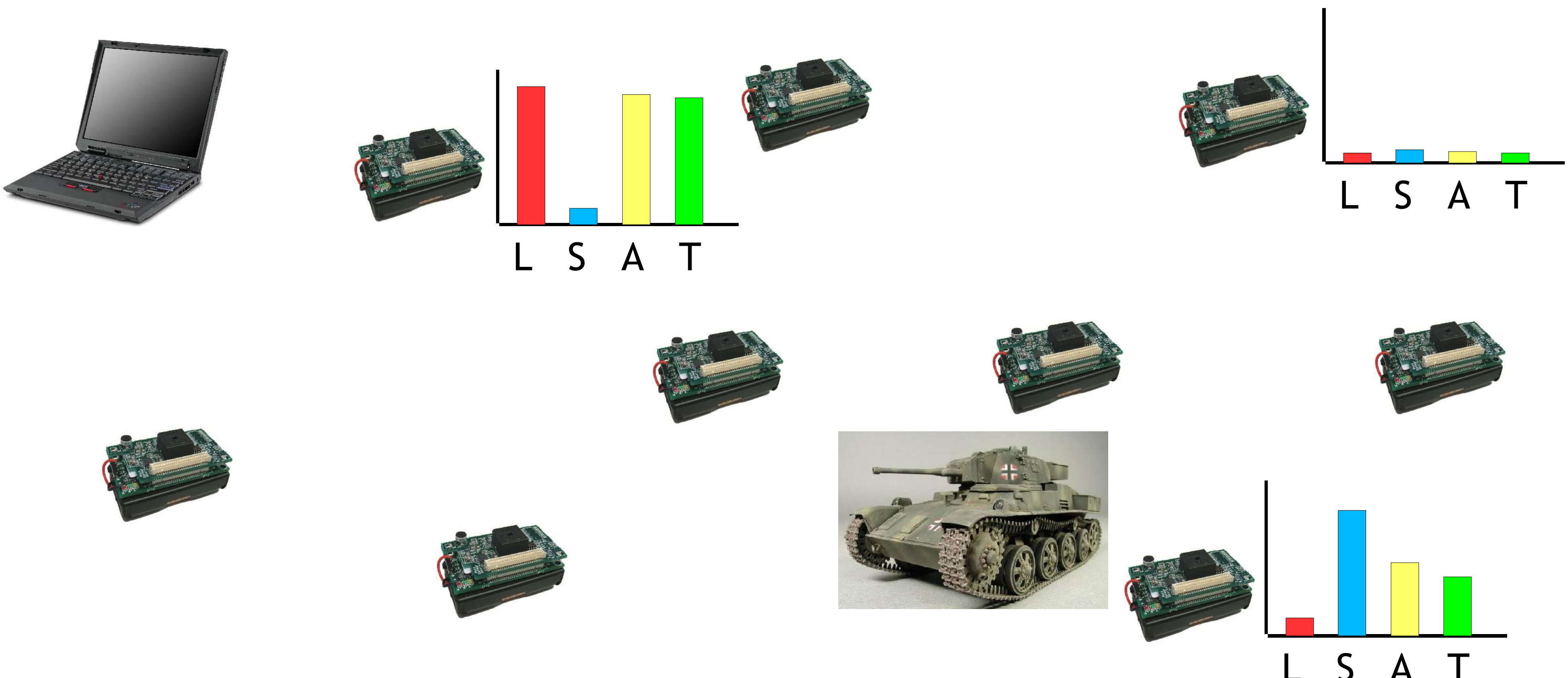
Nodes explore the action space to avoid falling into local minima

- ϵ-greedy reinforcement learning
- Nodes usually take the action with highest utility $u(a)$
- However, with (small) probability ϵ, the node takes a *random* action

Utility Function Example

Utility functions vary depending on node's position in network

- Nodes near target have high utility for sampling
- Nodes along routing path have high utilities for listening and sending



Listen Sample Aggregate Transmit

SORA Design Issues

Nodes operate using a very simple program

- Small amount of state and low computational overhead

Network rapidly adapts to changing conditions

- Nodes take actions that they individually find to earn rewards
- Learning strategy for utility function adapts behavior over time

No explicit coordination between nodes

- However, reward feedback leads to a natural equilibrium
- e.g., Fraction of nodes transmitting and listening for messages is *balanced*

Reward prices do not have a large impact on behavior!

- They only serve to differentiate behavior when multiple actions are profitable.
- For our experiments, we set the reward for each action to the same value
- More later...

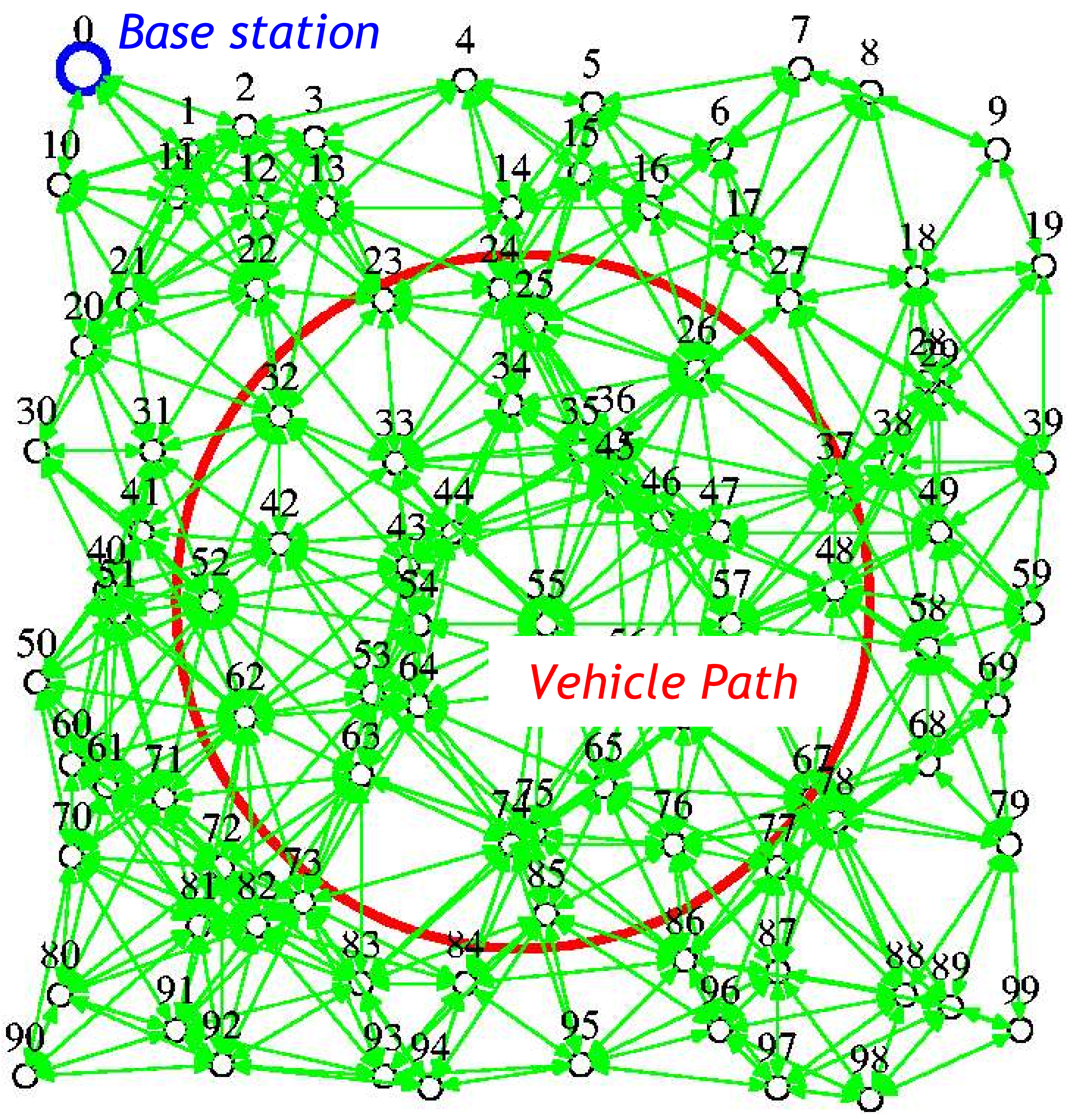
Evaluation Methodology

Simulation based on TinyOS environment

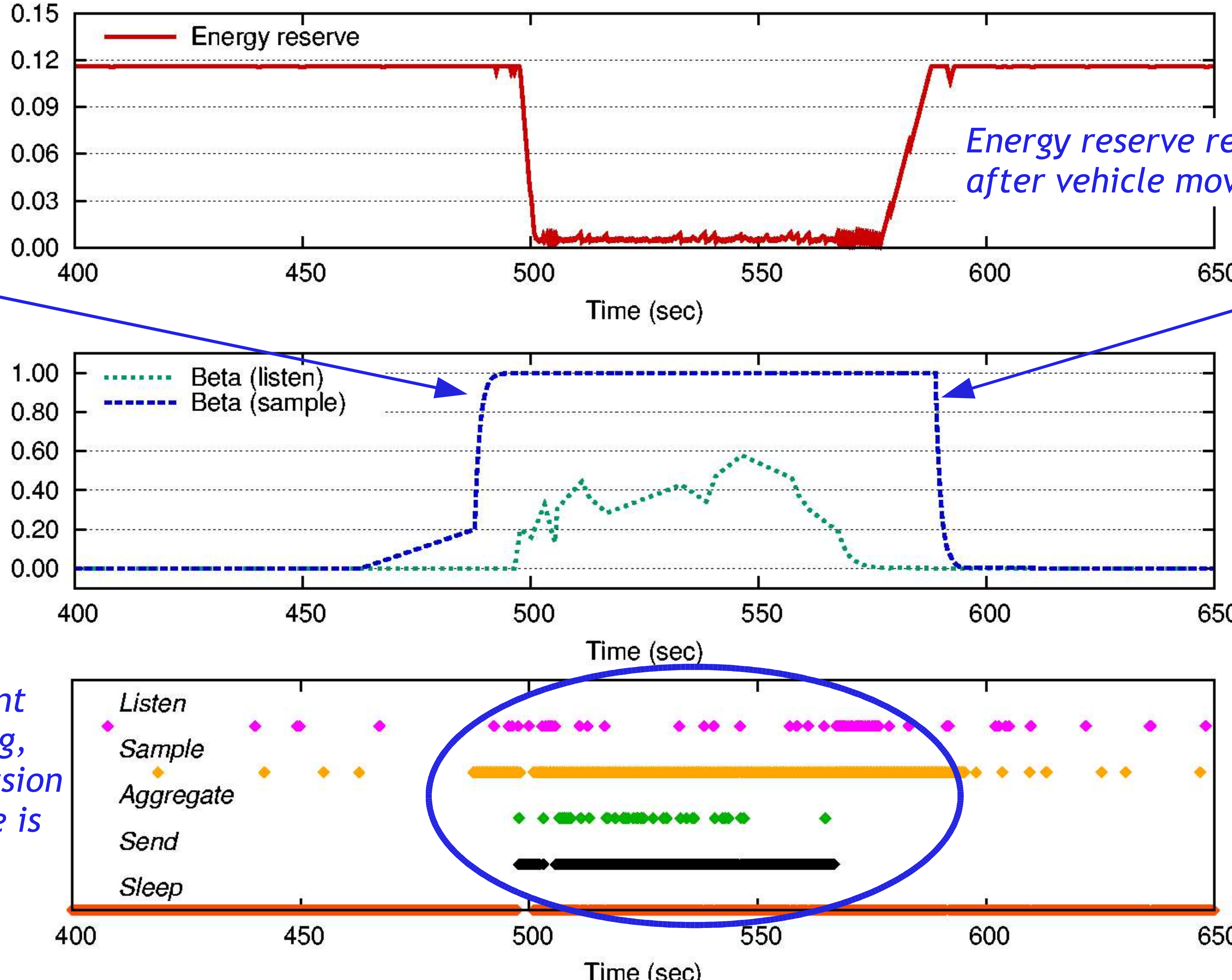
- Captures realistic hardware-level effects
- Target travels in circular path
- Routing using GPSR to base station

Evaluation goals:

- Can SORA achieve good tracking accuracy?
- How efficient is the resulting resource allocation?
- How well do nodes adapt to changing conditions?



Node Behavior over Time
(one node along the path of the vehicle)



Energy reserve refills after vehicle moves away

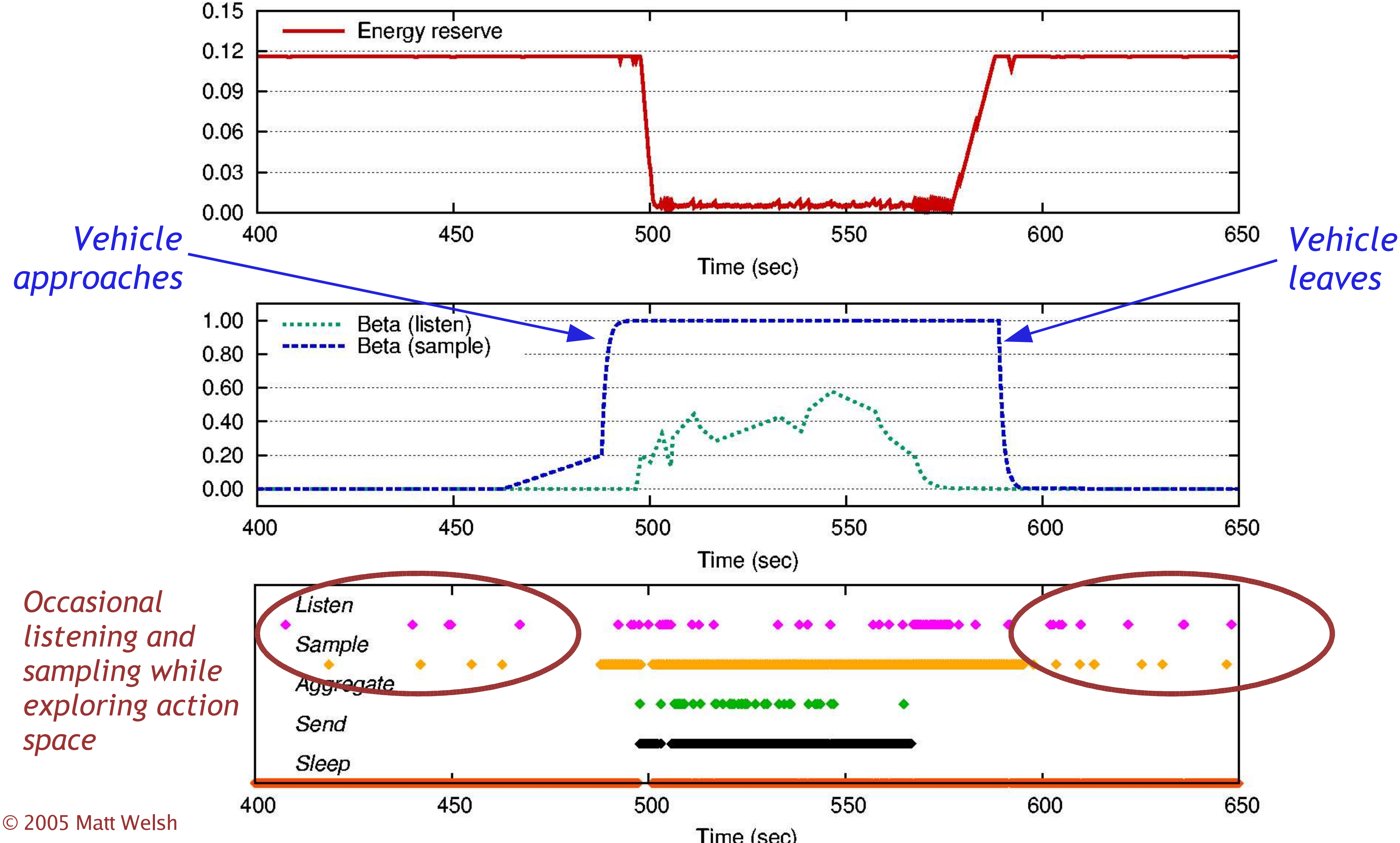
Vehicle approaches

Vehicle leaves

More frequent sampling, agg, and transmission while vehicle is nearby

Node Behavior over Time

(one node along the path of the vehicle)



Comparison to Alternatives

Implemented two alternative scheduling techniques

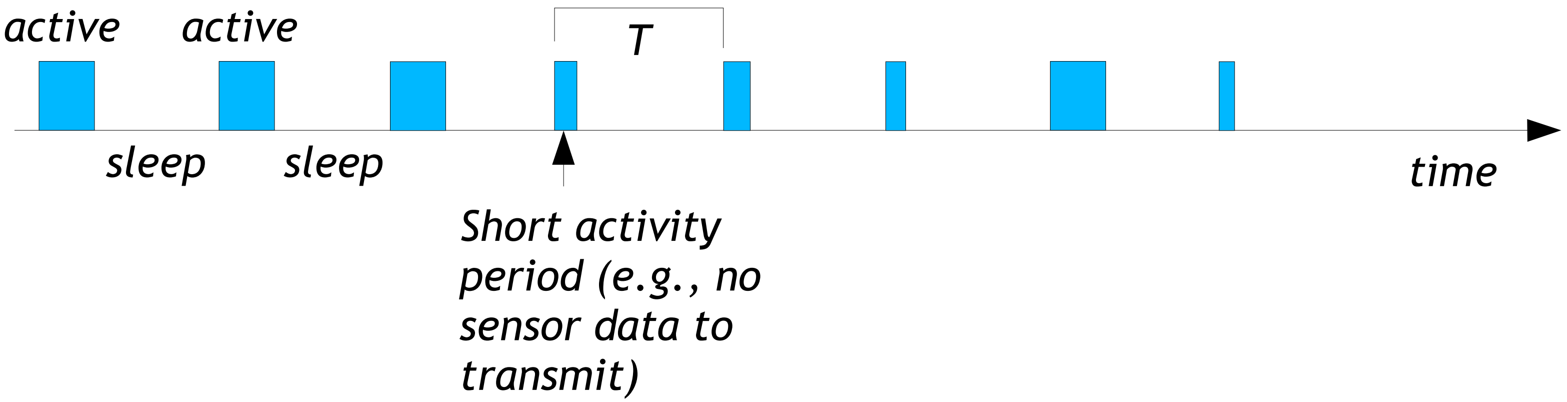
- (Plus a third described in the paper)

Each node is given a daily energy budget (e.g., 1000 J/day)

- Node's energy reserve *continually* refills at this rate

Static, periodic schedule (most commonly used technique today)

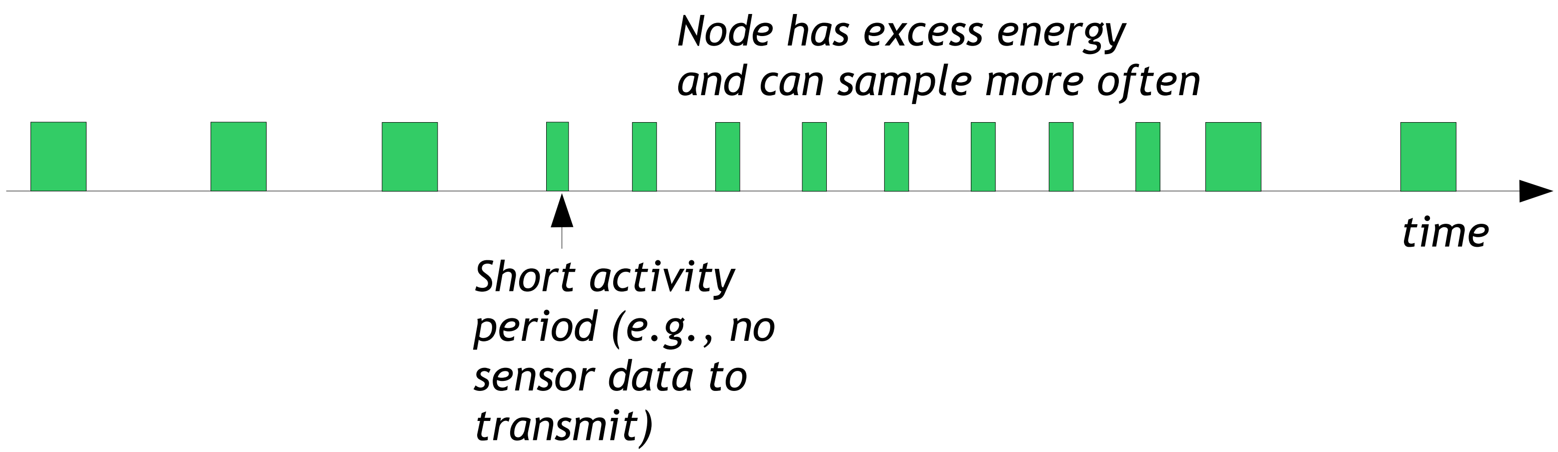
- Nodes periodically sample, listen, aggregate, transmit, and sleep
- All nodes operate at the same rate, calculated **offline** to meet energy budget
  - *This is conservative: Nodes may not use entire energy budget*



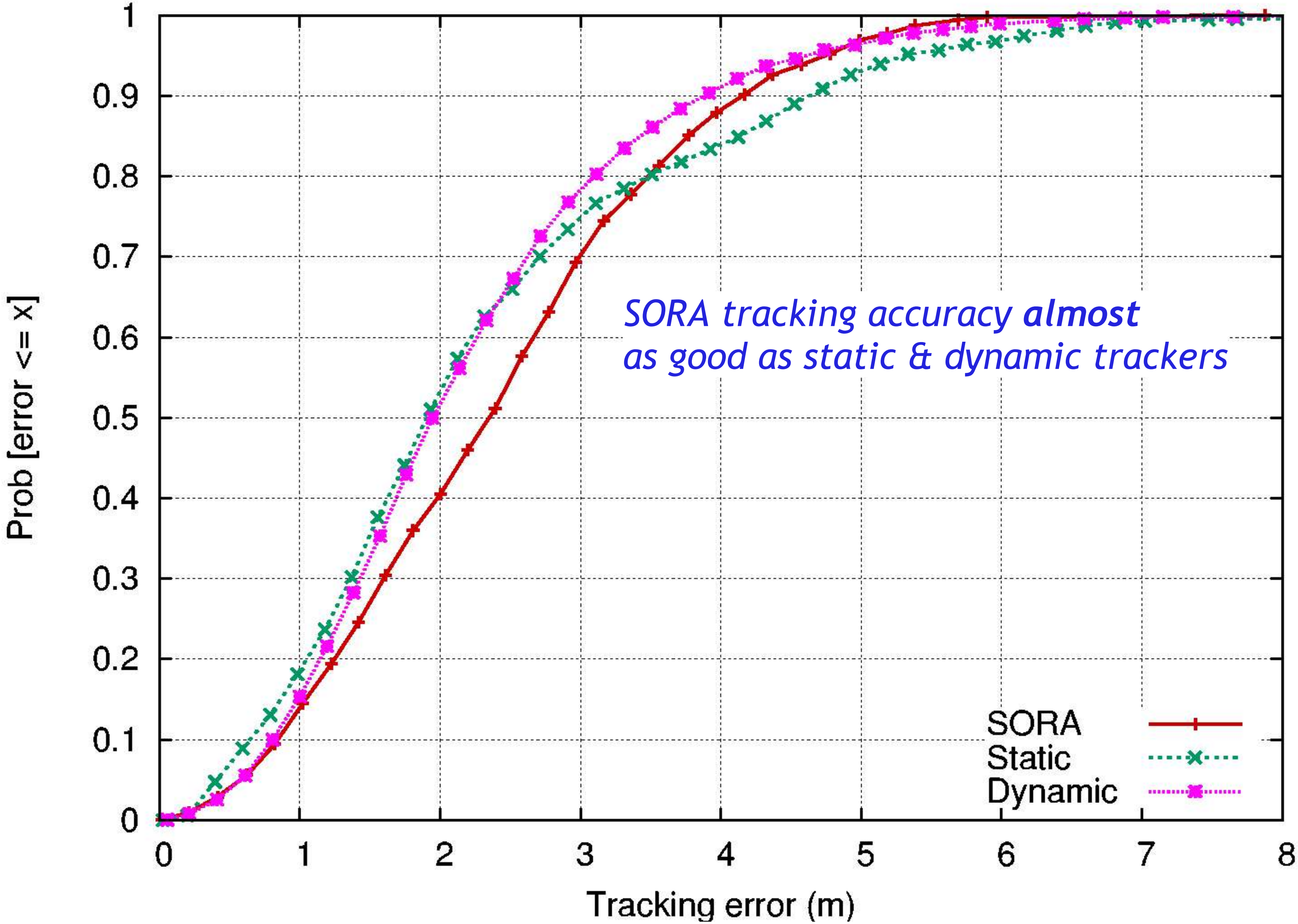
Comparison to Alternatives

Dynamic periodic schedule

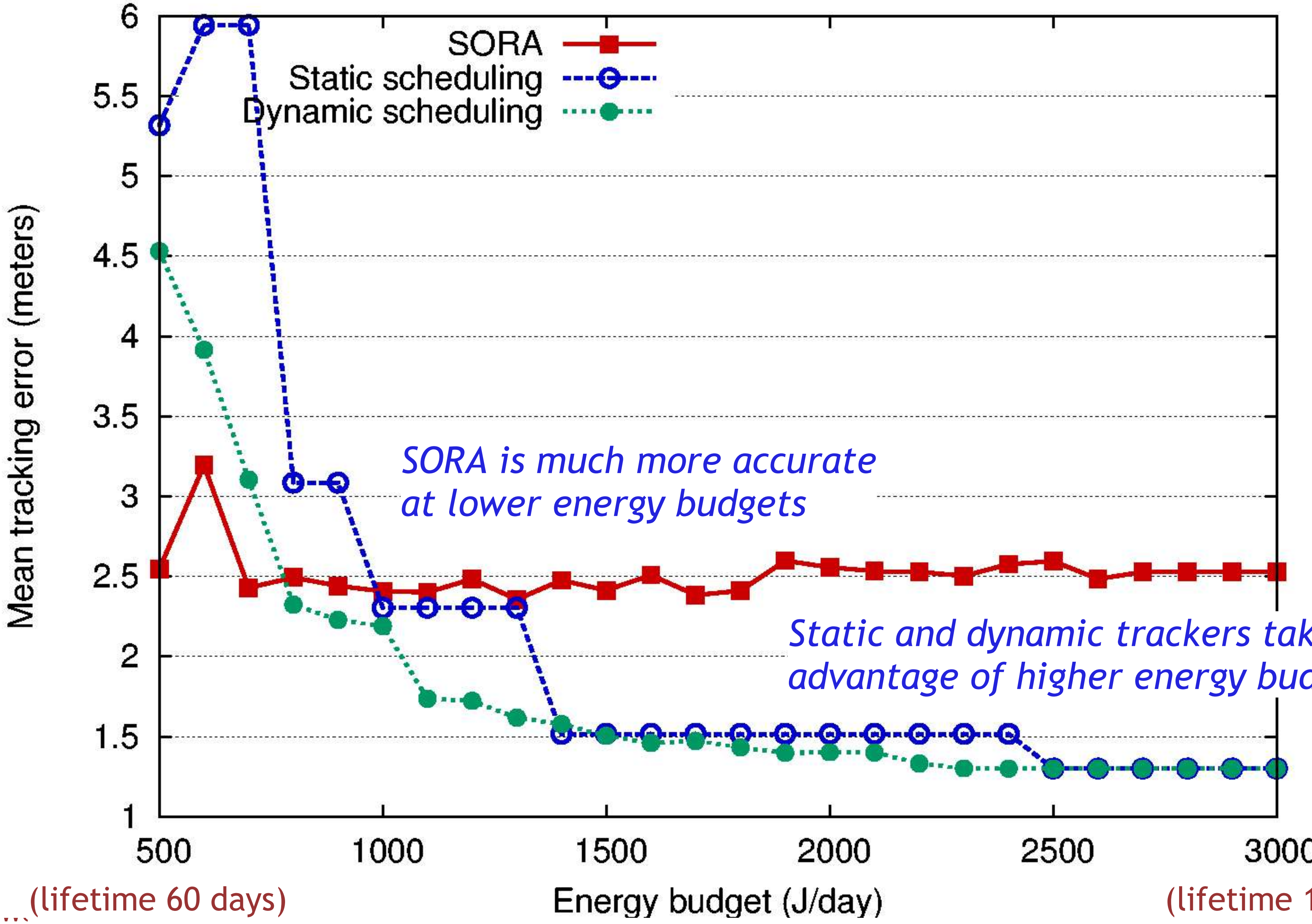
- Nodes dynamically tune processing rate to exactly exhaust their energy reserve
- Some nodes will operate at faster rates than others



Overall Tracking Accuracy
(energy budget 1000J/day)

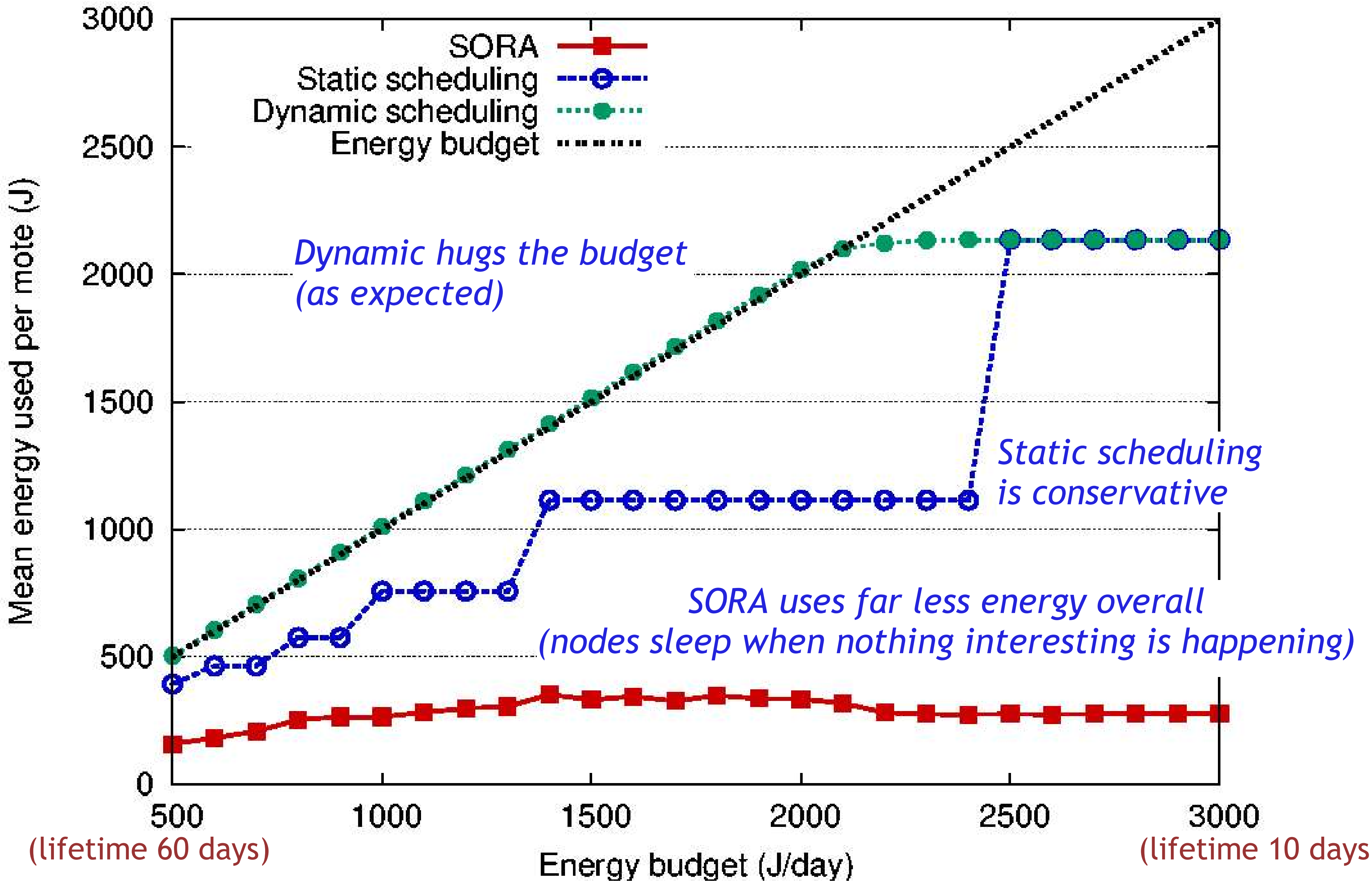


Effect of varying energy budget on accuracy



(lifetime 60 days) (lifetime 10 days)

Energy Use


SORA
Static scheduling
Dynamic scheduling
Energy budget
Dynamic hugs the budget
(as expected)
Static scheduling
is conservative
SORA uses far less energy overall
(nodes sleep when nothing interesting is happening)
Mean energy used per mote (J)
Energy budget (J/day)
0
500
1000
1500
2000
2500
3000
500
1000
1500
2000
2500
3000
(lifetime 60 days)
(lifetime 10 days)

Energy Efficiency

A key goal of SORA is to maximize the *efficiency* of the network:

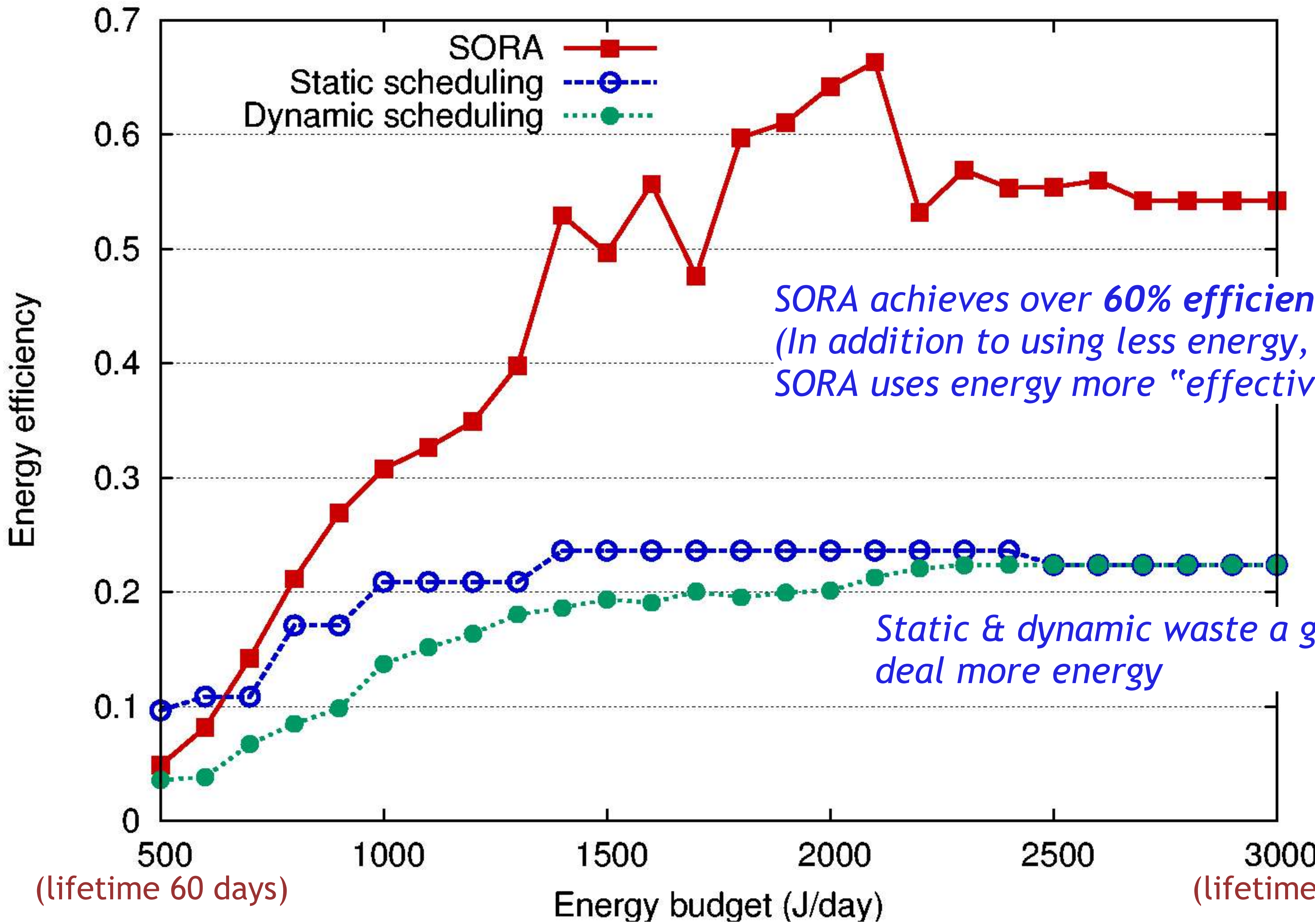
$$Efficiency = \frac{Total\ energy\ used\ to\ collect\ and\ route\ sensor\ data}{Total\ energy\ used\ by\ all\ nodes}$$

- We calculate this by tracking all energy use that resulted in a position estimate arriving at the base station.

Higher efficiency implies less wasted energy

- No realistic system can be 100% efficient
- Wasted energy due to taking bad sensor readings, listening at wrong times, etc.

Energy Efficiency



*SORA achieves over **60% efficiency**.*
(In addition to using less energy,
SORA uses energy more "effectively")

Static & dynamic waste a great deal more energy

(lifetime 60 days)

(lifetime 10 days)

What about varying prices?

We did extensive measurments with different rewards for each action.

Surprisingly, had little effect on tracking accuracy or energy use!

Observation:

- Prices only “matter” when a node has multiple actions with non-zero utility
- But a node can usually take only one action at a time!
  - *At least, this is the case in our tracking application.*

Most of the behavior in SORA is dictated by the learning process, not the choice of reward prices.

Also in the paper...

Experiments varying the learning parameters ϵ and α

- These impact the learning behavior and energy efficiency

Experiments using heterogeneous energy budgets

- Give some nodes a large energy budget (e.g., connected to mains power)

Experiments with non-uniform reward settings

- Configure some nodes as “routers” and others as “sensors”

Future Directions

Allow nodes to reason about future opportunities for profit

- Current scheme very myopic: Nodes always pick most profitable action
- Would like to price valuable *sequences* of actions
  - *e.g., Must sample multiple times before aggregating*

Extend model to allocate resources across multiple users

- Each network user can pay for different sets of actions
- Use *equilibrium pricing* to seek Pareto optimal resource allocations

Use reward settings to retask sensor nodes on the fly

- e.g., Nodes on the edge of the network can act as “sentries” detecting vehicle arrival
- Interior nodes can stay dormant
- When sentry detects vehicle, floods a new reward vector to retask interior nodes

Conclusions

Sensor networks need new tools for managing resources

- Energy and bandwidth are very constrained
- Manual scheduling and allocation is difficult to get right

Our approach: Self-Organizing Resource Allocation (SORA)

- Decentralized, adaptive scheduling of individual node operations
- Nodes use reinforcement learning to tune their behavior over time

SORA achieves:

- High tracking accuracy (nearly as good as “static” scheduling techniques)
- Very low energy usage (nodes learn when to activate on short time scales)
- High energy efficiency (little wasted energy taking useless actions)

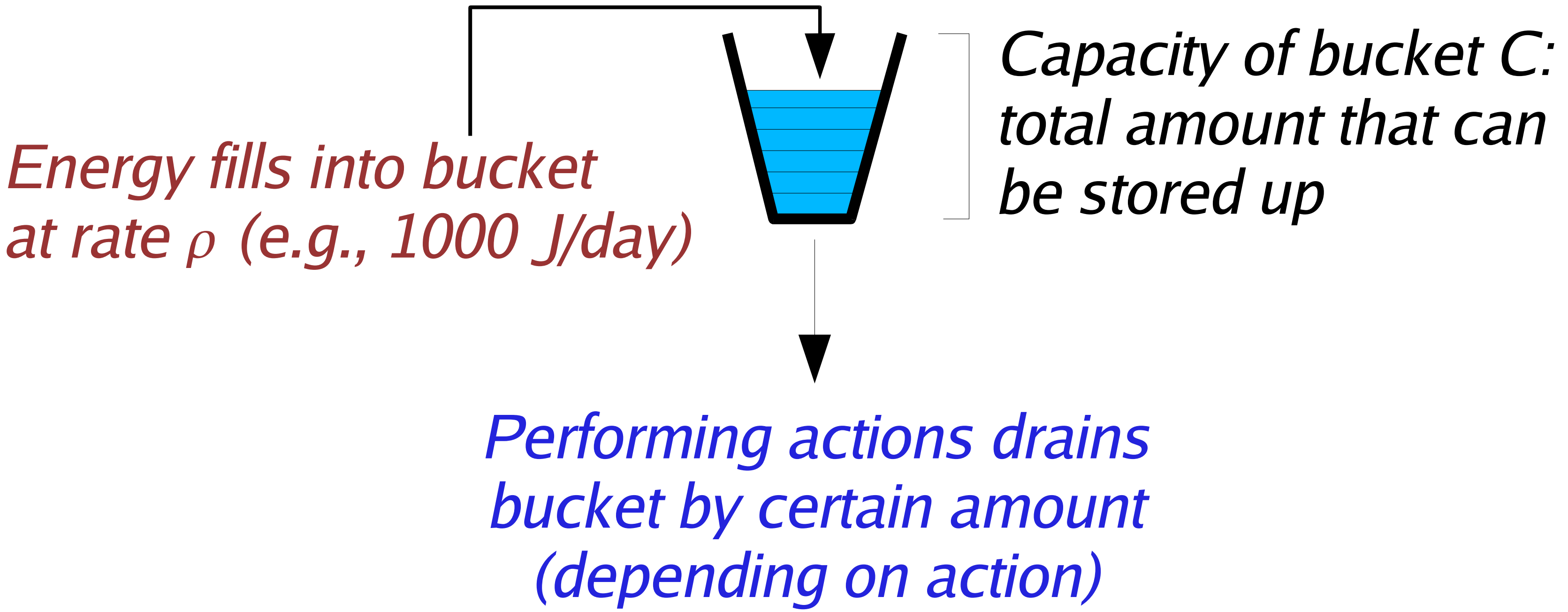
http://www.eecs.harvard.edu/~mdw/proj/mp

Energy Budget

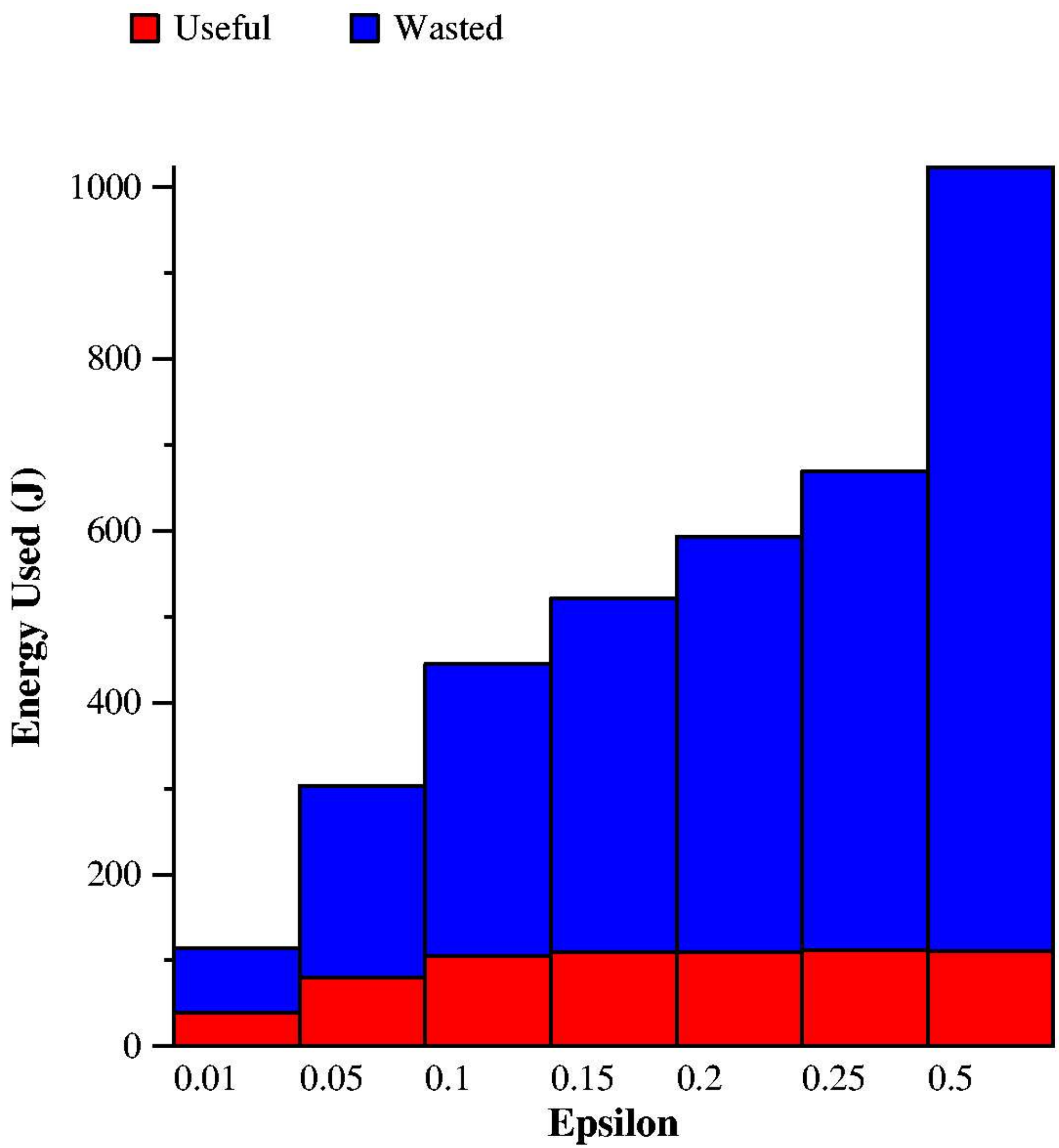
Most important constraint on node operation is energy

We model the energy budget for each node as a *token bucket*

- Rate of bucket fill determines average rate of energy use
- Capacity of bucket bounds “burst size”



Effect of varying exploration probability



ϵ parameter determines how often node selects a random action

- Low ϵ: Node usually chooses highest-utility action
- High ϵ: Allows node to find new profit faster

Low ϵ: most energy wasted taking high-utility (but not useful!) actions

High ϵ: most energy wasted exploring the action space

Best setting seems to be somewhere in the middle

Actions and energy cost

Nodes can select from four actions:

Sample the magnetometer (84 μJ)

- Results in sample value that scales with distance to vehicle
- Cannot detect vehicle if more than 11 m away

Listen for incoming radio messages (5.9 mJ)

Aggregate multiple accumulated readings (84 μJ)

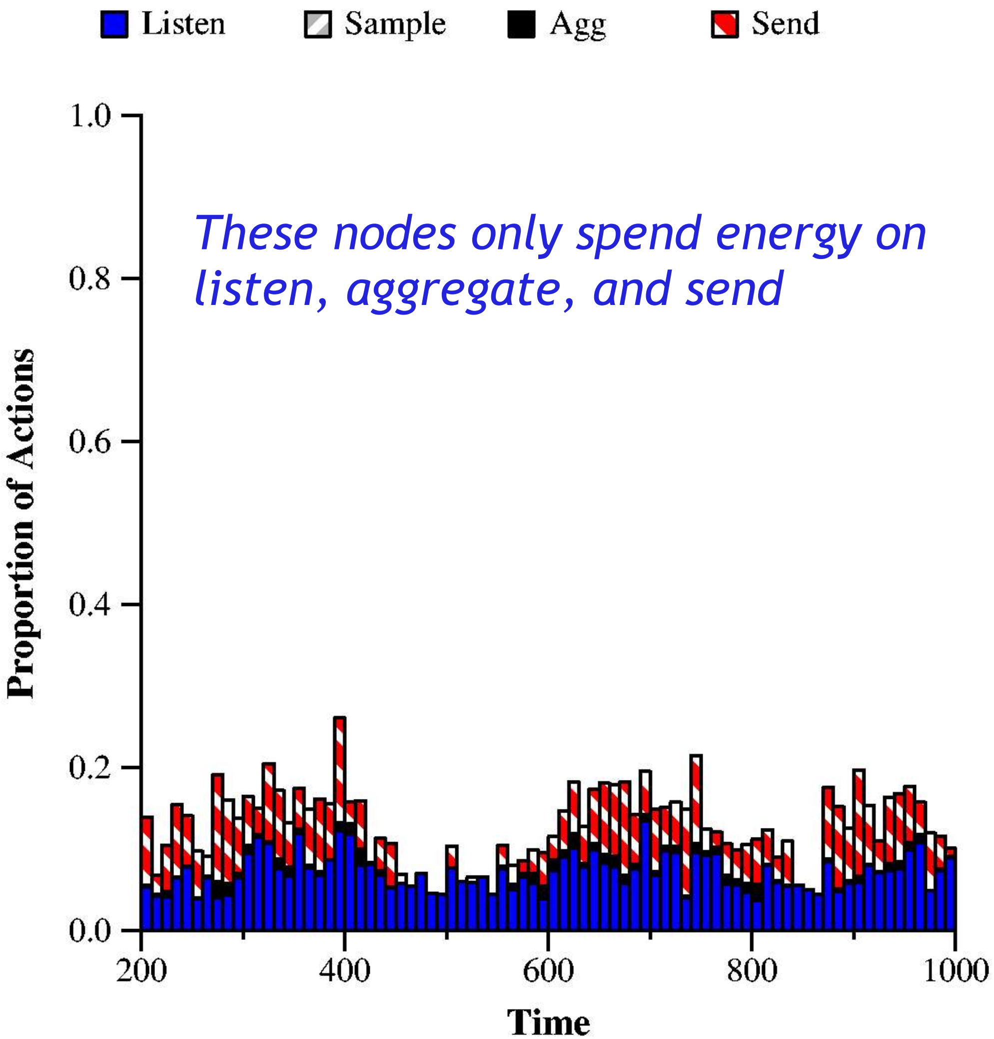
- Computes partial centroid of accumulated values

Transmit a message towards the base station (2.4 mJ)

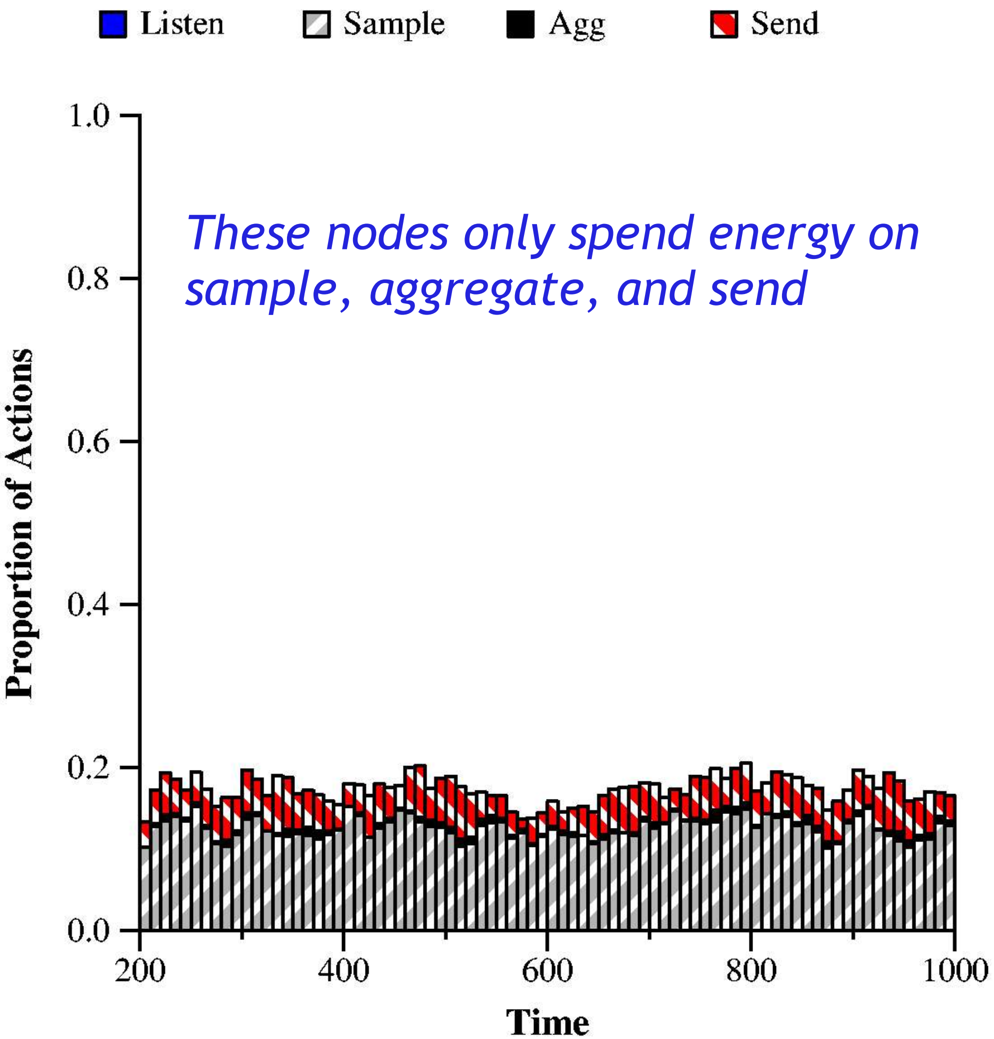
- Uses GPSR routing
- Any node closer to the base that is currently *listening* will receive the message

Effect of Prices on Action Choice

“Routers”
(20% of nodes)



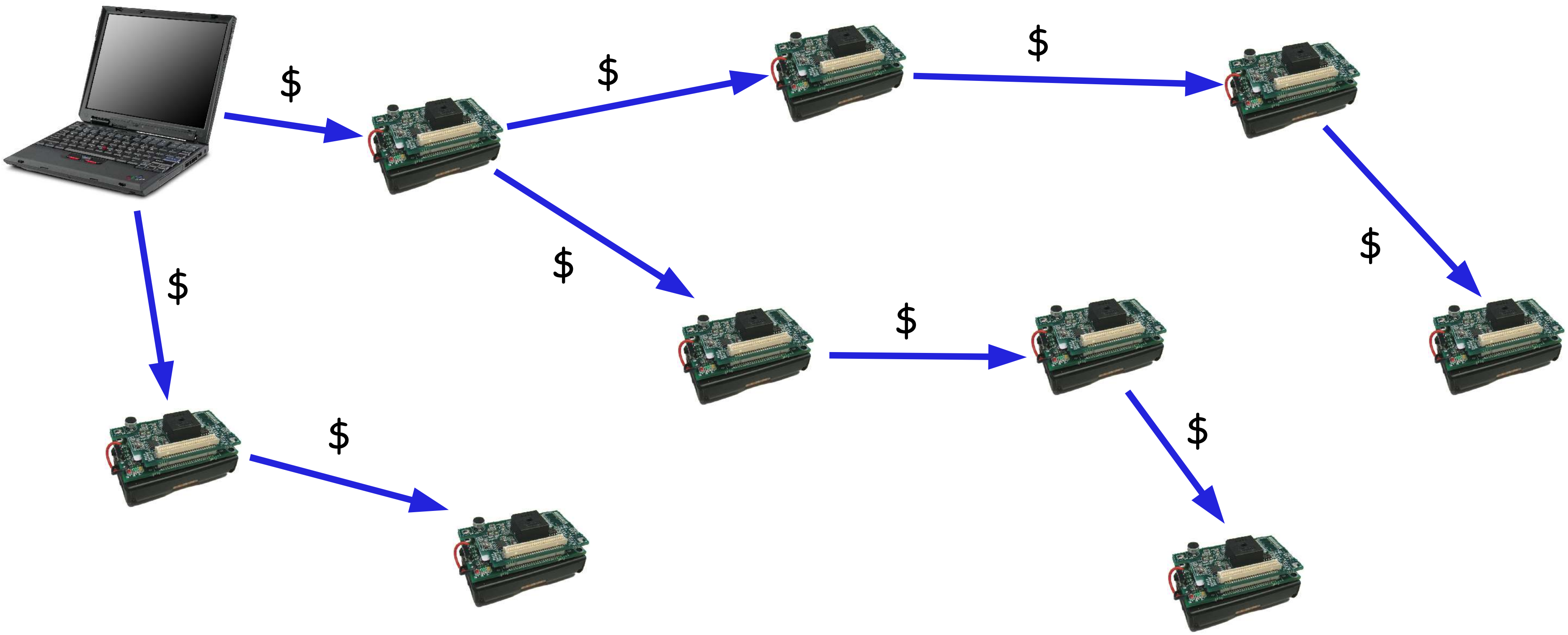
“Sensors”
(remaining 80%)



Price Propagation

First step: Flood prices for each good to the network

- Several efficient protocols for this (e.g., Trickle)
- Can readily update prices on the fly



Comparison to Alternatives

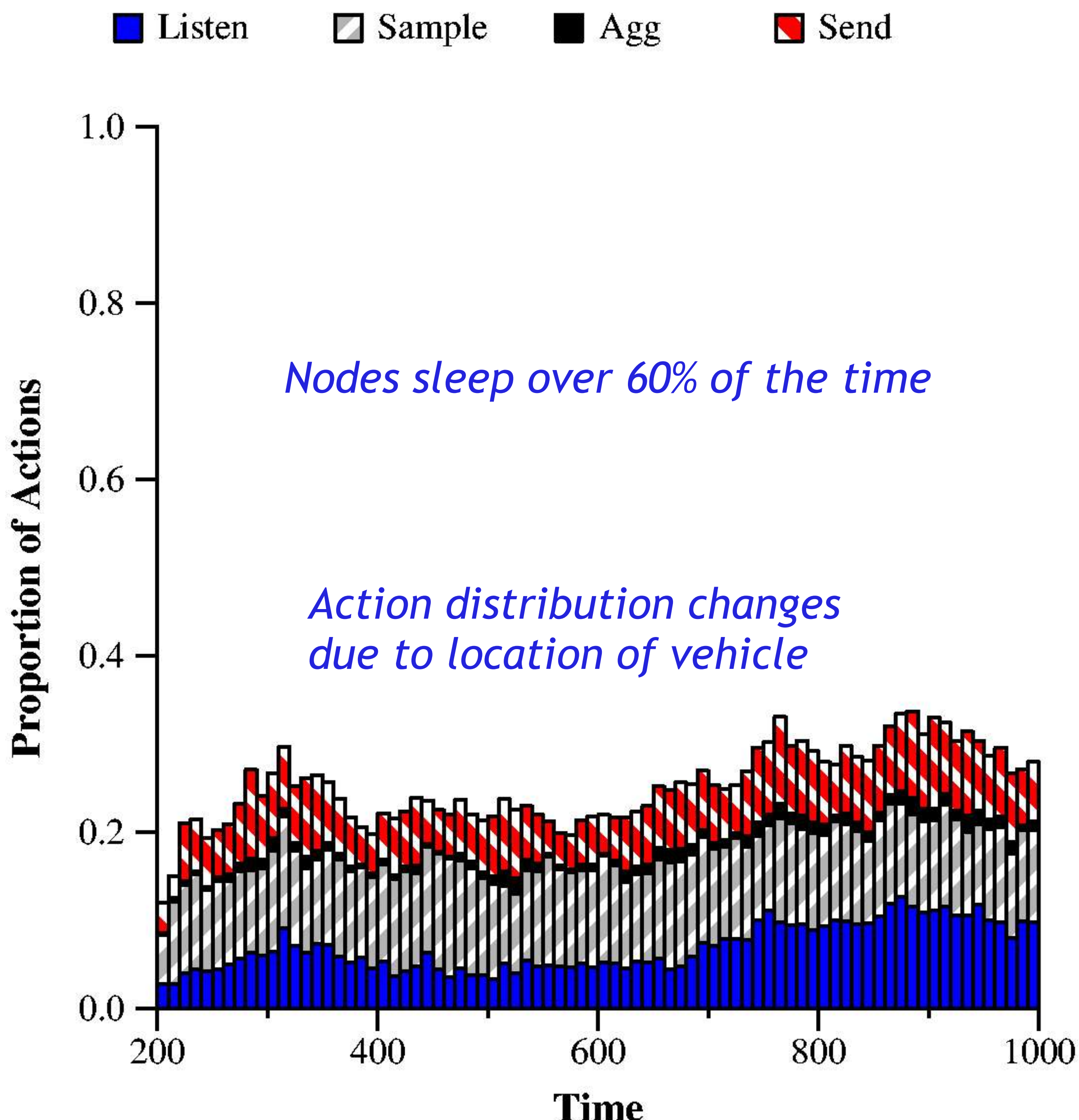
Hoods [Whitehouse et al., MobiSys'04]

- Programming abstraction for neighborhood-based communication

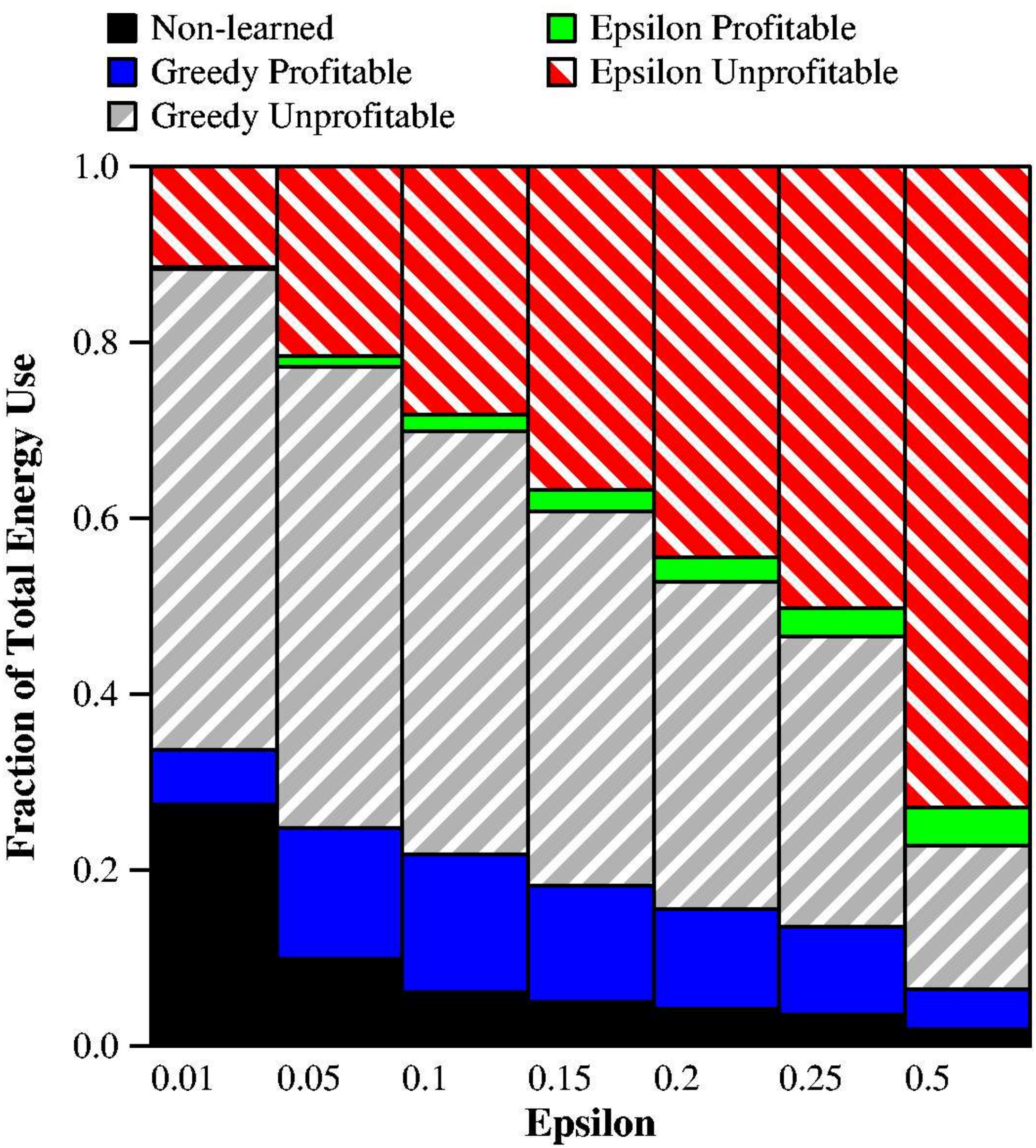
Hoods implements a different approach to tracking:

- Nodes broadcast sensor values to neighborhood
- “Leader” node aggregates data and sends position estimate to base station
- We found that this is less accurate than the SORA, static, and dynamic trackers

Actions taken over time



Effect of varying exploration probability



ϵ parameter determines how often node selects a random action

- Low ϵ: Node usually chooses highest-utility action
- High ϵ: Allows node to find new profit faster

Low ϵ: most energy wasted taking high-utility (but not useful!) actions

High ϵ: most energy wasted exploring the action space

Best setting seems to be somewhere in the middle