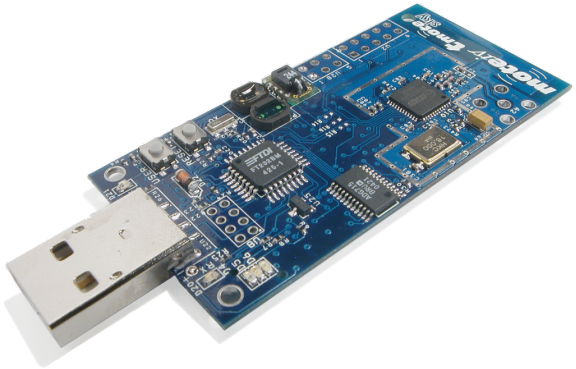
Peloton:
Coordinated Resource Management for Sensor Networks

Jason Waterman, Geoffrey Werner Challen, and Matt Welsh

Harvard
School of Engineering
and Applied Sciences

Sensor Networks: Coping with Limited Resources

- Sensor nodes are *severely* resource constrained
  - 8 MHz CPU
  - 10 KB of memory
  - ~100 Kbps of radio link bandwidth (best case)
  - 200 mAh – 2000 mAh batteries



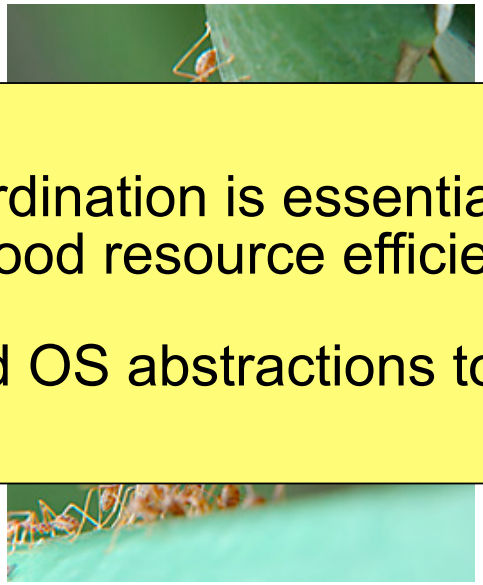
Z-MAC
Dozer
S-MAC
MAC Layer
T-MAC
B-MAC

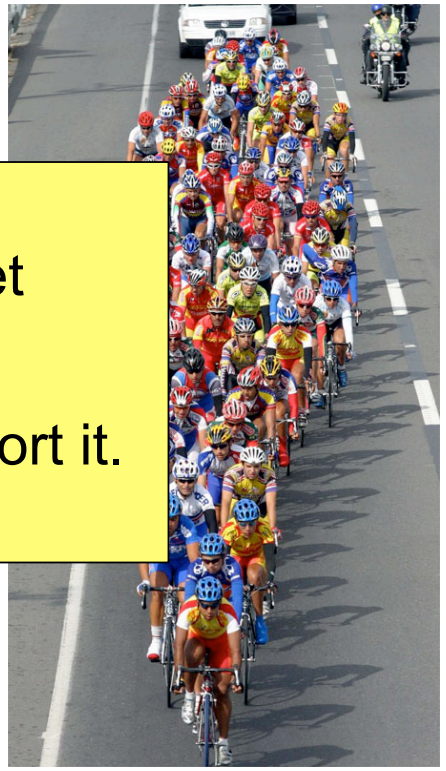
Pixie
Eon
OS and
Programming models
Levels
Power
Locks

Quantro
Energy
Tracking
PowerTOSSIM

Focus on optimizing
at the node level

Coordination Matters





Coordination is essential to get good resource efficiency.

We need OS abstractions to support it.

State of the Art

Conventional Distributed Systems

- Map/Reduce
- DHTs
- BFT
- BPEL
- Group communication
- Web services
- XML RPC
- JINI
- Discovery services
- IDLs
- Multicast
- RPC
- BSD sockets
- TCP/IP

Sensor Networks

Everything else is done in an ad hoc manner by each application.

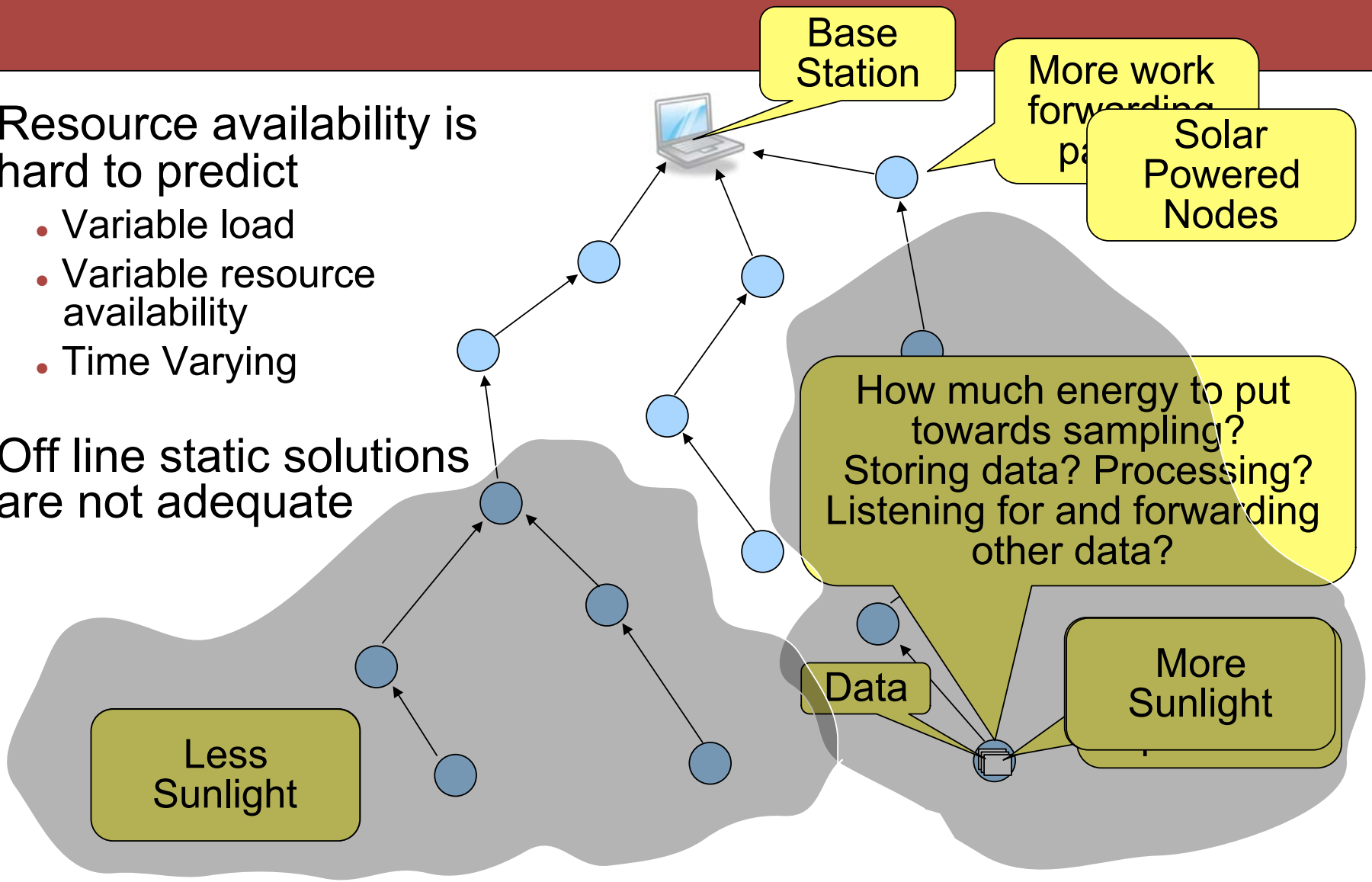
- Active Messages
- Radio packets

A Canonical Example: Data Collection

Resource availability is hard to predict

- Variable load
- Variable resource availability
- Time Varying

Off line static solutions are not adequate



Programming with Resource Management

The three keys to effective coordinated resource management

- Sharing resource availability across nodes
- Exposing resource availability to the application
- A way of allocating resources across multiple nodes

Currently, applications do this in an ad hoc fashion

- Point solutions tied to a specific application
- Requires lots of engineering
- Hard to tune

The Peloton Operating System

Distributed OS for wireless sensor networks that provides coordinated resource management

Three key ideas:

- State Sharing – allows for nodes to share information local resource availability
- Vector Tickets – represents the right to consume resources across a set of nodes for performing an operation
- Ticket Agents – permits resource management decisions to be decomposed across the network

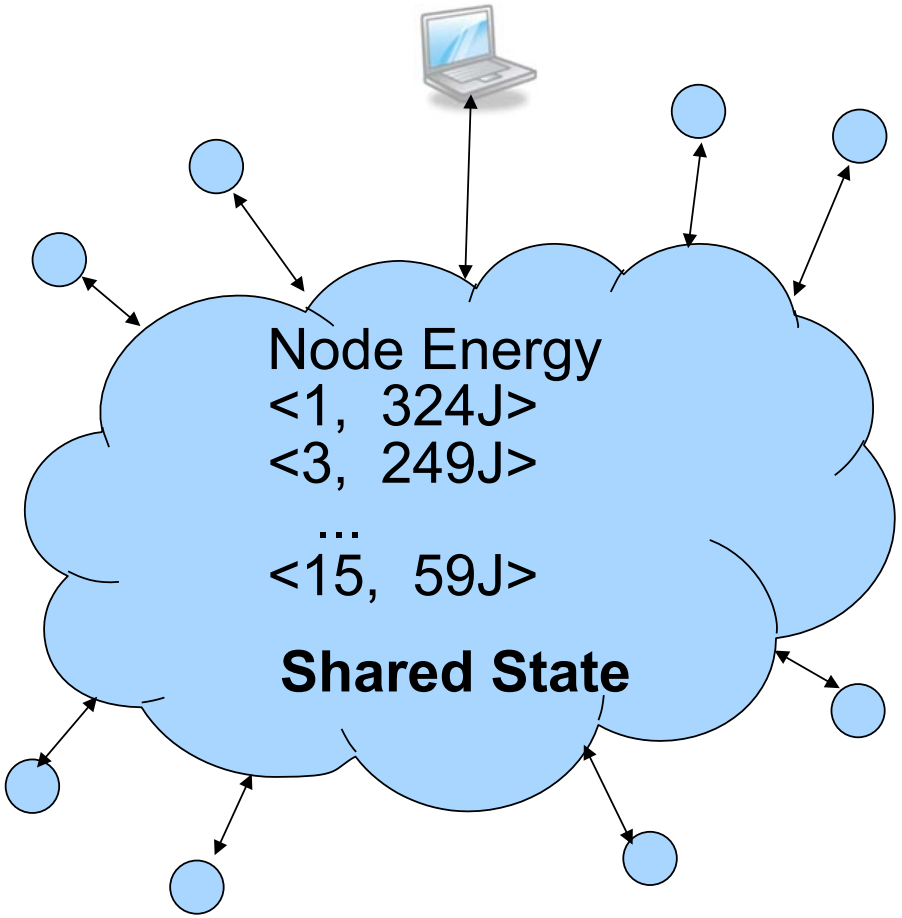
Key Idea #1: Efficient State Sharing

Shared tuple space

- Representing a global view of the network state
- Builds on the work of Abstract Regions [NSDI '04] and Hood [MOBISYS '04]

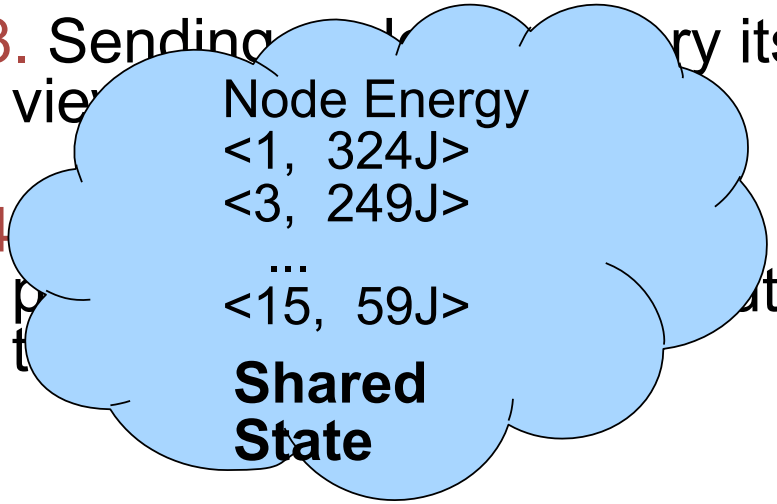
Efficient Implementation

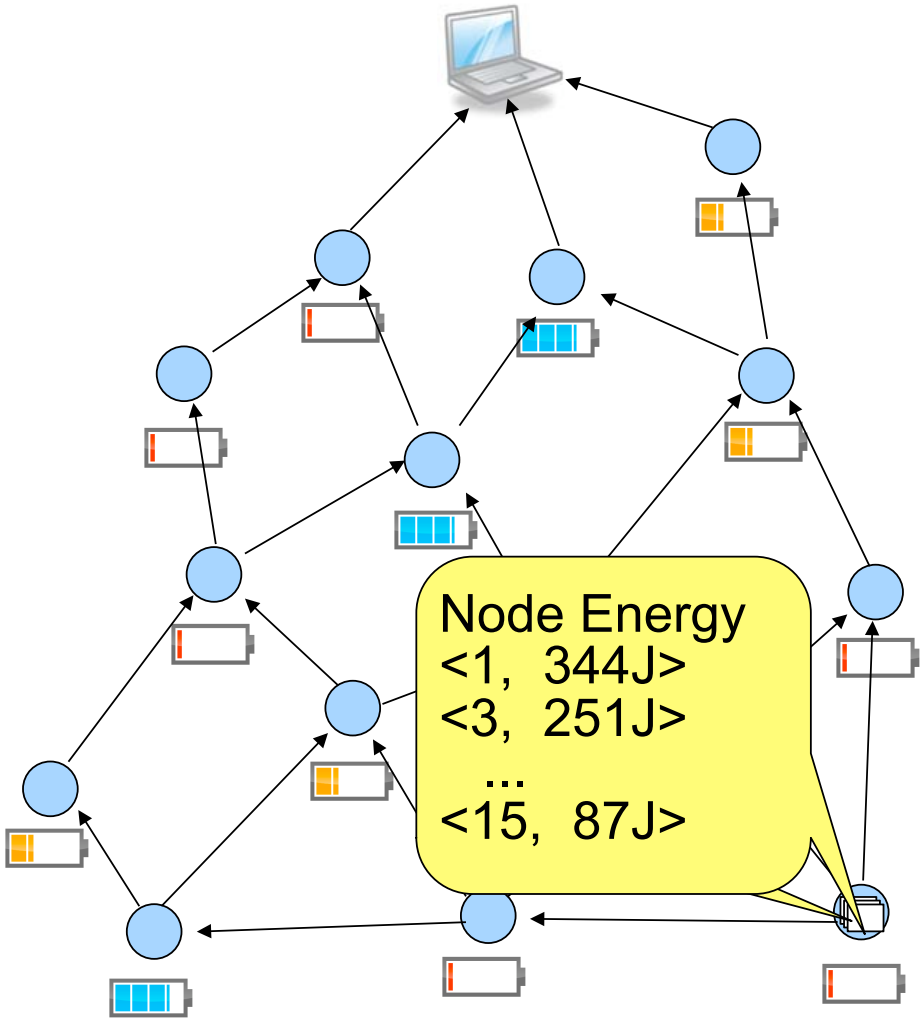
- State on other nodes is read only – no write conflicts
- Use gossip to ensure eventual consistency
- Topological Freshness – the frequency of updates between two nodes is related to the distance between them



Data Collection with State Sharing

1. Nodes track their own state
2. Nodes periodically publish their state into the shared tuple space
3. Sending ... ry its vie...
4. ... ate p... t...





Representing Resources

Problem: Existing systems track resources implicitly

- No direct feedback to applications on availability
- Allocation and use of resources are tightly coupled

Need a mechanism to represent allocation across a set of nodes

- e.g., Amount of energy/memory/bandwidth used to route data to the base station

The Pixie Resource-Aware OS

[Sensys'08]

Node-level OS for sensor nets based on the concept of **resource aware programming**

- Resources as a first-class programming primitive
- Direct application knowledge of resource availability
- Explicit allocation and revocation of resources: TANSTAAFL principle

Key idea: **Resource tickets**

- Ticket **<R, c, t_e>** represents right to consume **c** units of resource **R** until the expiry time **t_e**.
- Think of as a short-term "reservation" for some resource.

Tickets provide...

- Direct visibility over resources
- Fine-grained control
- Rich abstraction for adapting to changing conditions

Key Idea #2: Vector Tickets

Peloton extends Pixie's model to span allocations on multiple nodes

- **Vector Ticket** = < T_1, T_2, T_3, ...T_n >
- T_n = ticket for resources at node **n**

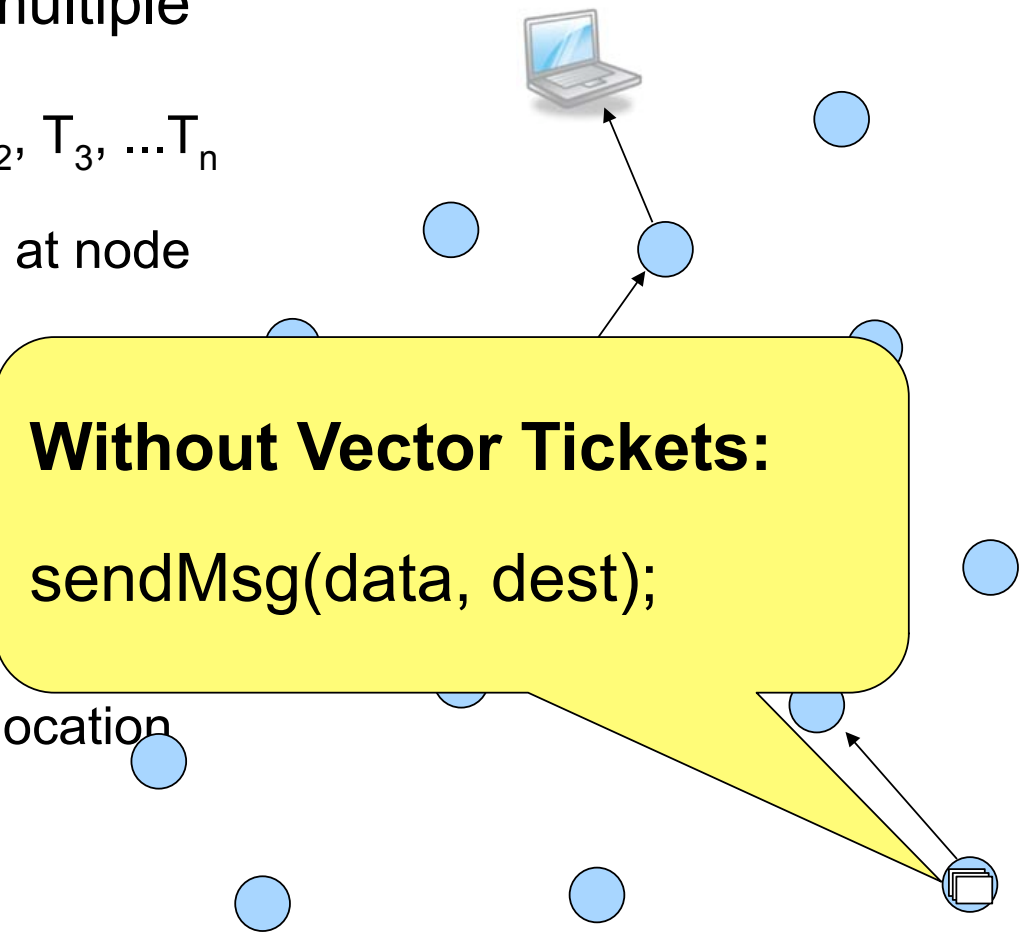
What do vector tickets l

- A global accounting me
  for resources

  We can account for all r

  allocation

Like ordering off a menu without any prices



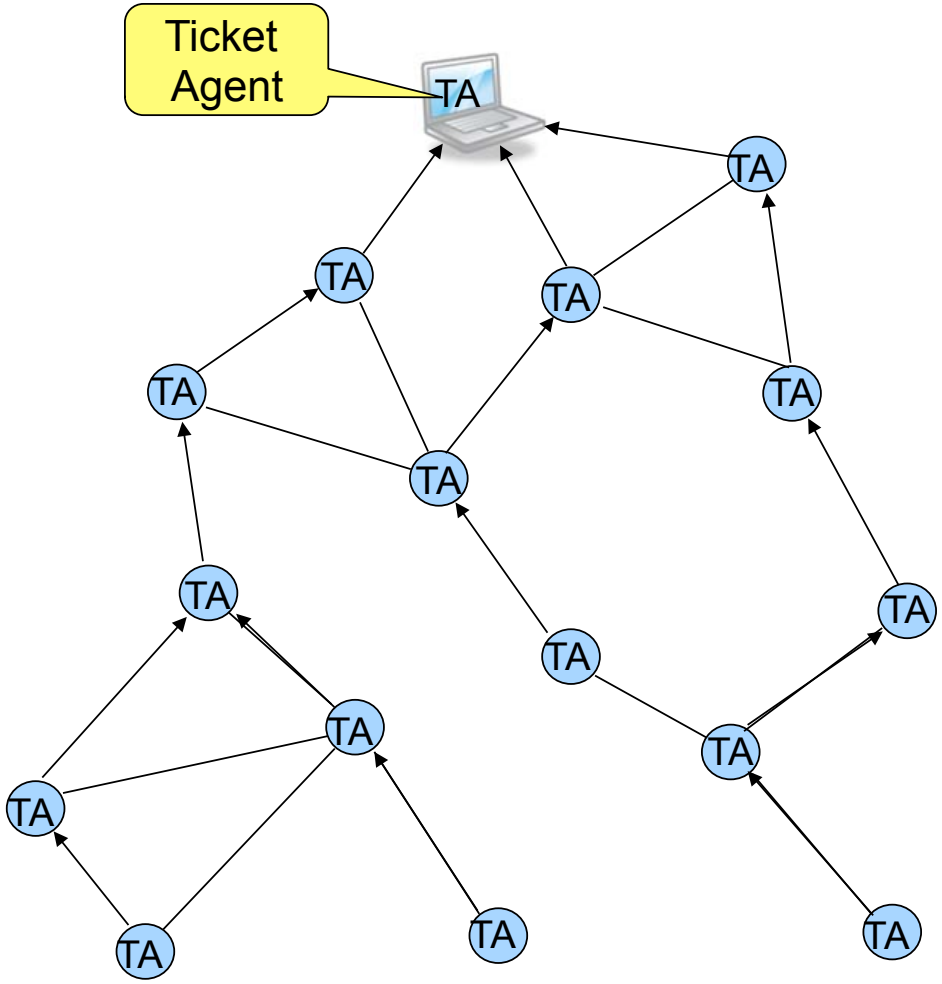
Key Idea #3: Who Allocates the Vector Tickets?

Resource allocation policies are provided by **ticket agents**

- Consume information from shared state
- Allocate vector tickets for operations in the network

Ticket agents can be implemented in multiple ways:

- Centralized
- Decentralized
- Cluster based



Application Vignettes

In the paper we highlight the Peloton architecture through three use cases that leverage the programming model

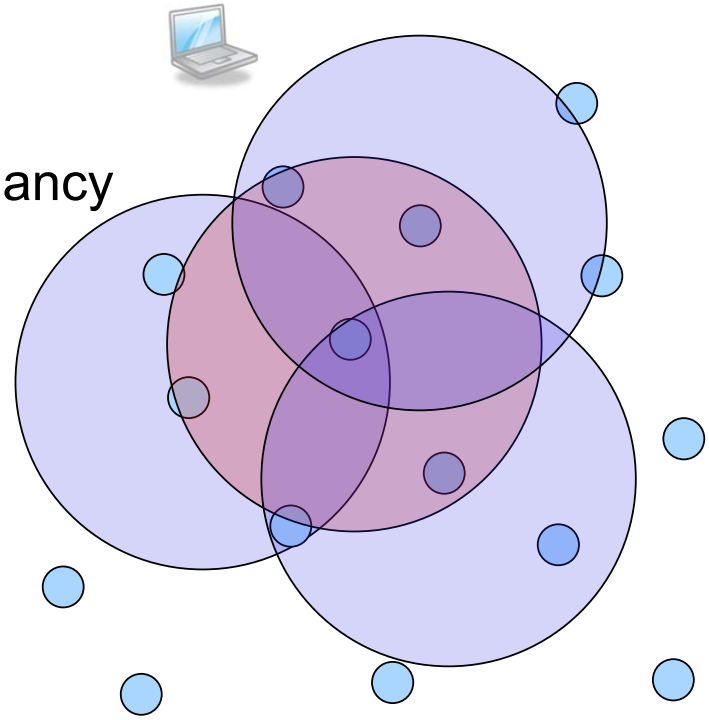
1) Adaptive sensor duty cycling

2) Adaptive cluster-based routing

3) Energy-efficient data collection

Adaptive Duty Cycling for Sensor Networks
[IPSN'04]

Used in surveillance where nodes detect events within their given sensor range



Assumptions:

- Dense network deployment to allow redundancy in sensor coverage
- Nodes know their own location

Naive approach: random duty cycling

- Can result in loss sensor coverage
- Non-adaptive to changes in resources

Coordinated duty cycling

- Higher overhead
- Tune sleep and wake cycles of nodes to maintain sensor coverage across network
- Adapt to variations in energy availability

Challenges in Coordinated Duty Cycling

Nodes must share their location and coverage information and learn the location and coverage of other nodes

Nodes need to know which nodes they are covering and for how long to cover them

- Can't sleep while covering for someone else

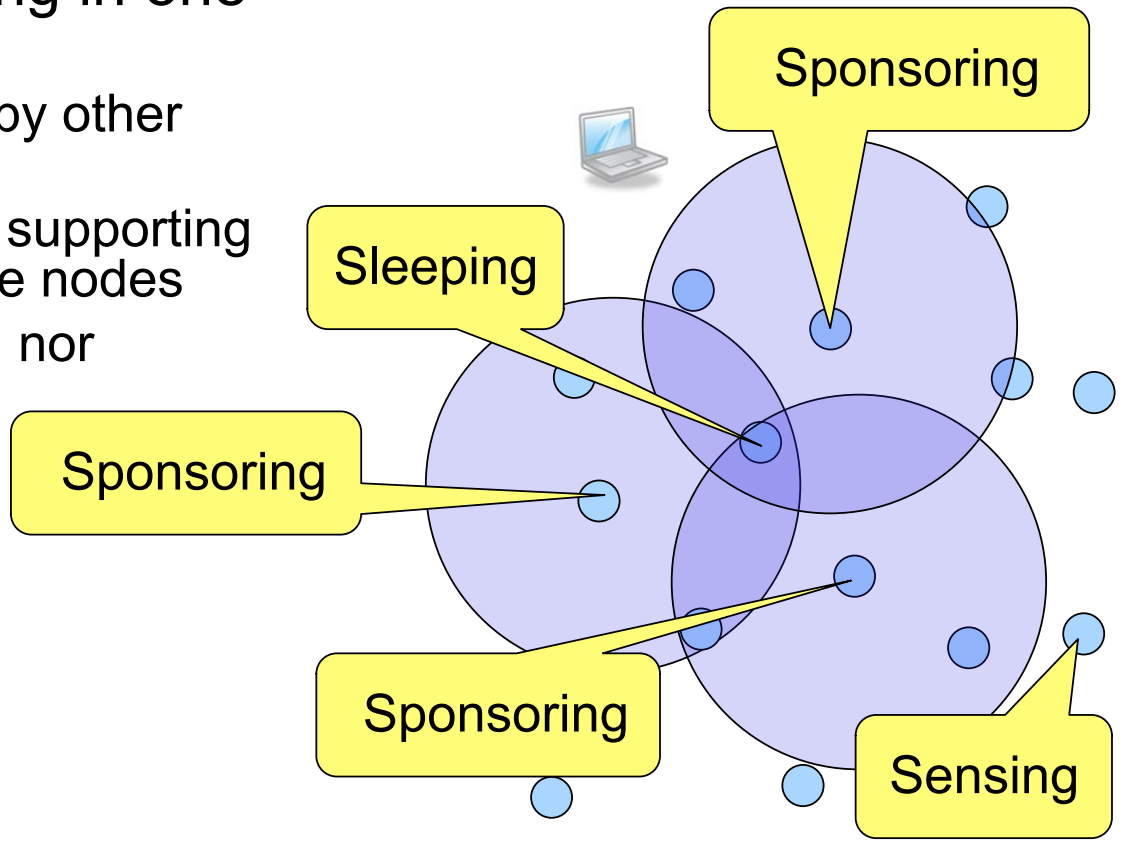
How to coordinate sleep schedules such that energy remains balanced in the network?

A Role-Alternating, Coverage-P reserving, Coordinated Sleep Algorithm (RACP)

RACP is a role based system

A node is always perorming in one of three roles:

- Sleeping – being covered by other nodes
- Sponsoring – sensing and supporting the sleeping of one or more nodes
- Sensing – neither sleeping nor supporting other nodes



A Role-Alternating, Coverage-P reserving, Coordinated Sleep Algorithm (RACP)

Nodes periodically send status packets with location, current role, and residual energy of the node

Sensing nodes periodically check for coverage sponsors

- If enough potential sponsors are found, ask for sponsorship with a random delay based on current energy
- If sponsors agree to the request, sleep, otherwise try again later

In Peloton, this is trivial to implement:

- Location, current role, and energy are published to the shared tuple space
  - *Topological freshness works well here as nodes only care about the state of their immediate neighbors*
- When a suitable set of sponsors have been found, the node's ticket agent generates a vector ticket containing resources for all the sponsors
- When the sponsors' ticket agents receive the vector ticket, they now have the resources needed to sponsor the node and begin to sponsor it and the sponsored node can sleep

Current Status

The design of Peloton has been motivated by the challenges we faced in our previous volcano sensor network deployments

- Tungurahua in '04 [EWSN '05], Reventador in '05 [OSDI '06], and Tungurahua again in 07 [SenSys '08]
- High data rates and load fluctuations from the environment made careful resource management essential to success

New goal: in-network signal processing and perpetual sensor deployments, running on top of Peloton OS

- Earthquake localization from seismic sensor data
- Solar powered nodes; even greater resource variability

Volcano monitoring is a challenging, real-world problem well suited to coordinated resource management

- Work has just begun
- Targeting a 2010 deployment

Conclusions

Existing approaches to managing resources in sensor nets are:

- Hard to tune
- Inefficient
- Based on local greedy heuristics

What we are proposing: SOCIALISM for our sensor networks

The three tenets of the Peloton manifesto are:

- State sharing
- Vector tickets
- Ticket agents

Thank you!

waterman@eecs.harvard.edu